Segmentation and Clustering

COS 429: Computer Vision

PRINCETON UNIVERSITY

Acknowledgments: T. Funkhouser, K. Grauman, S. Lazebnik, S. Seitz, X. Ren

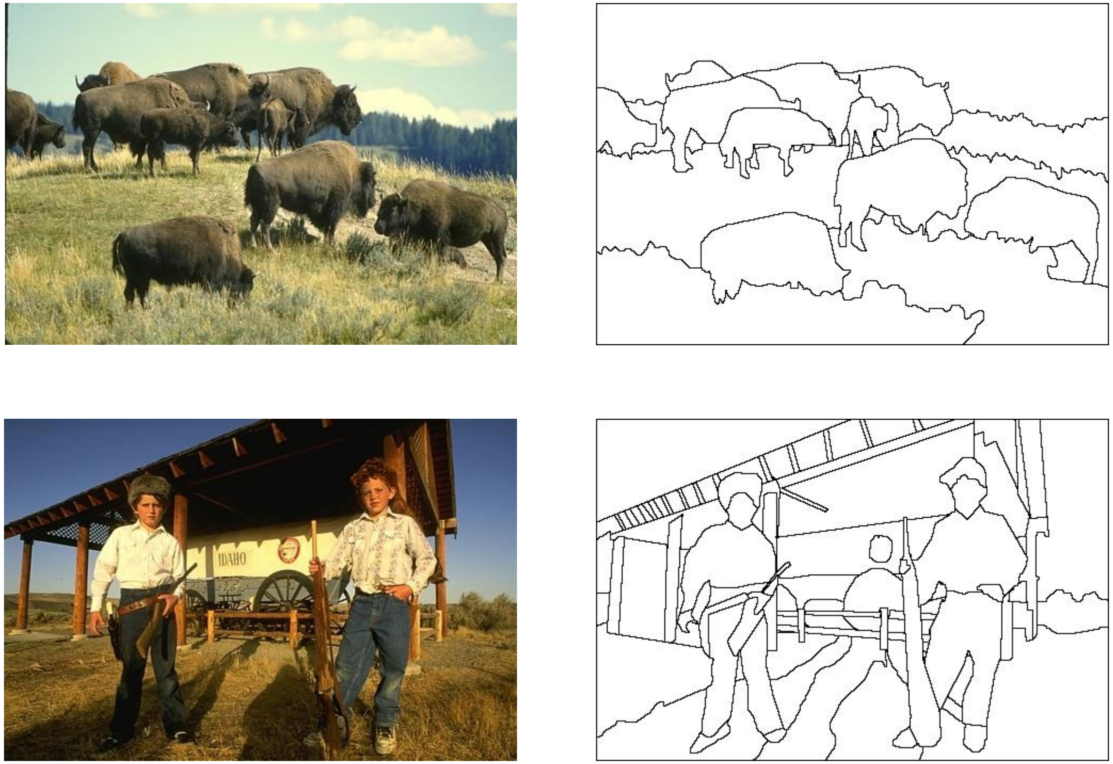
Segmentation and Clustering

- Segmentation:
  Divide image
  into regions
  of similar contents

- Clustering:
  Aggregate pixels
  into regions
  of similar contents

Goal

- Separate image into coherent “regions”



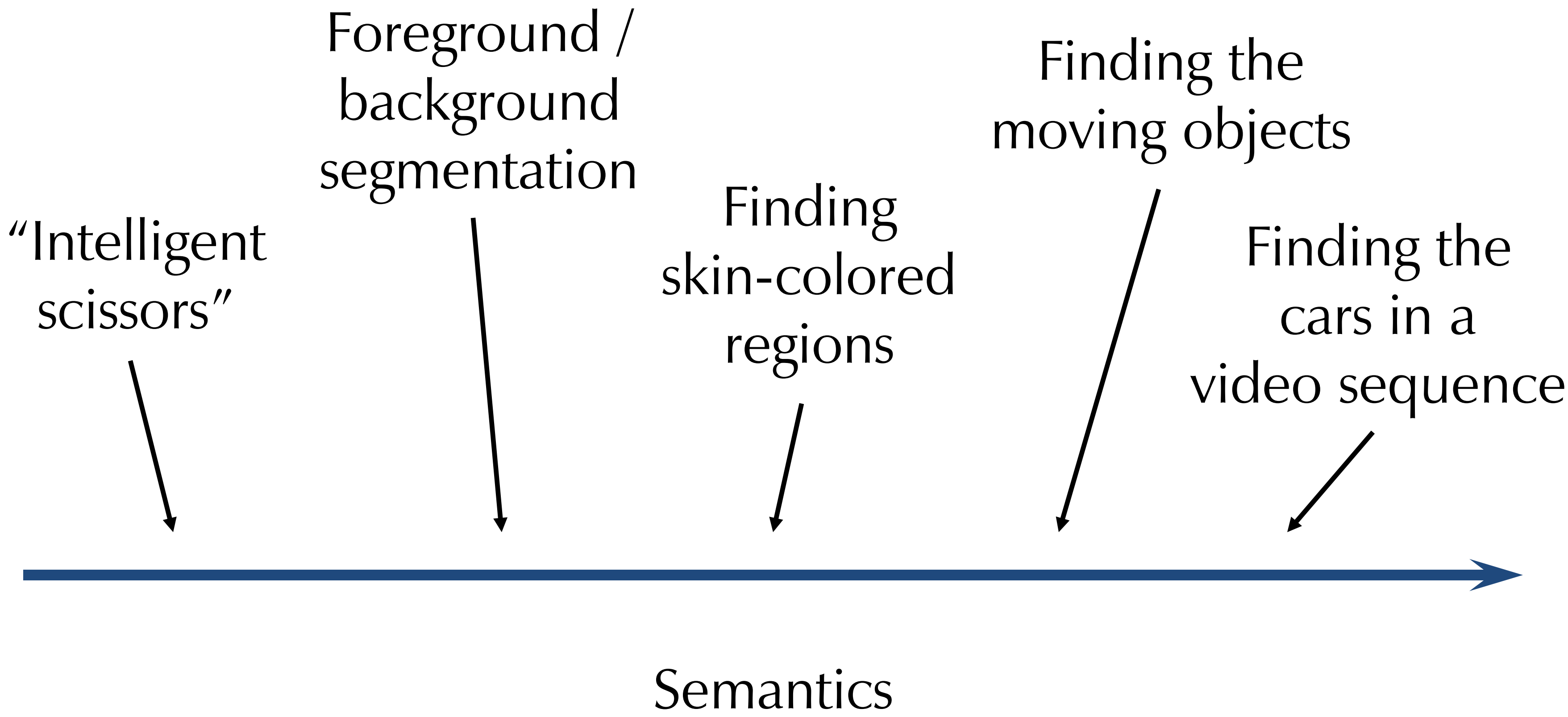
Berkeley segmentation database:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Lazebnik

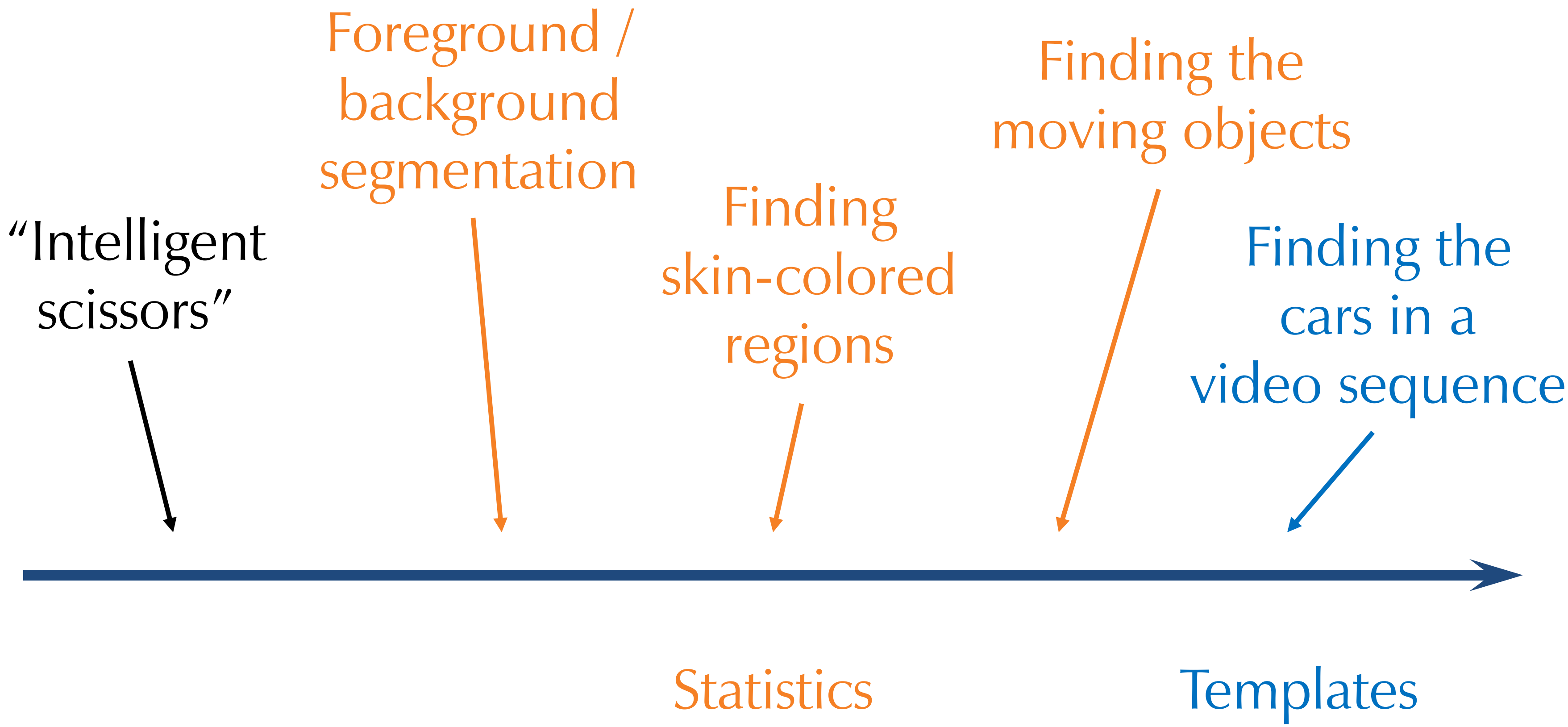
But Wait!

- We speak of “segmenting” foreground from background
- Segmenting out skin colors
- Segmenting out the moving person
- How do these relate to “similar regions”?

Segmentation and Clustering Applications



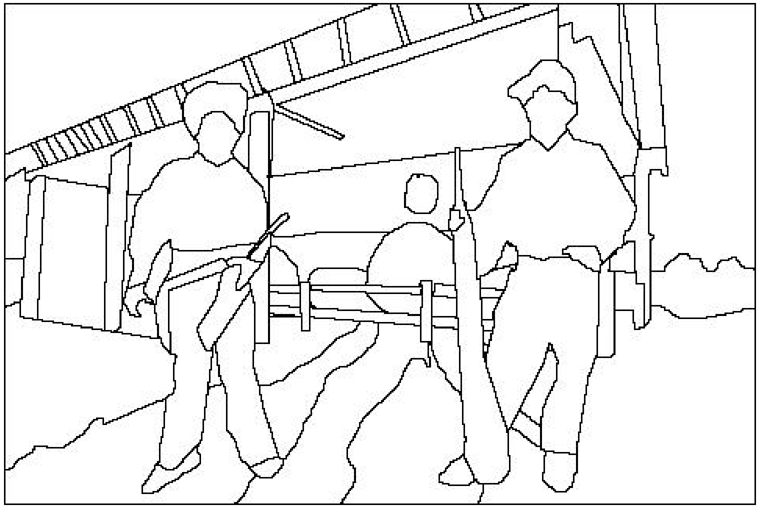
Segmentation and Clustering Applications



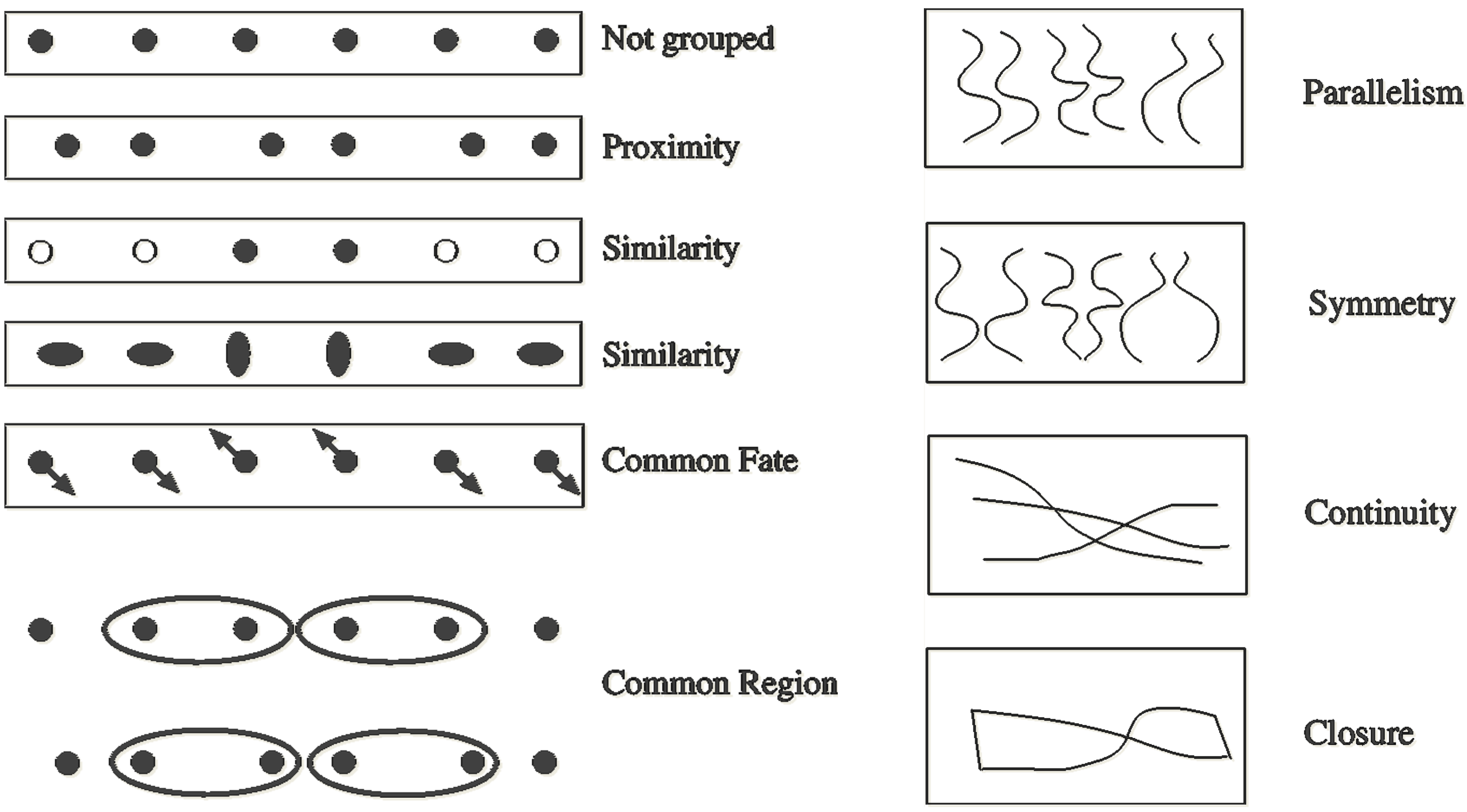
Questions

- What is coherent?
  - Similar color?
  - Similar texture?
  - Spatial proximity?

- What kinds of regions?
  - Nearly convex?
  - Smooth boundaries?
  - Nearly equal sizes?
  - What granularity?



Gestalt Grouping Cues



Segmentation and Clustering

- Defining regions
  - Should they be compact? Smooth boundary?
- Defining similarity
  - Color, texture, motion, …
- Defining similarity of regions
  - Minimum distance, mean, maximum

Clustering Based on Color

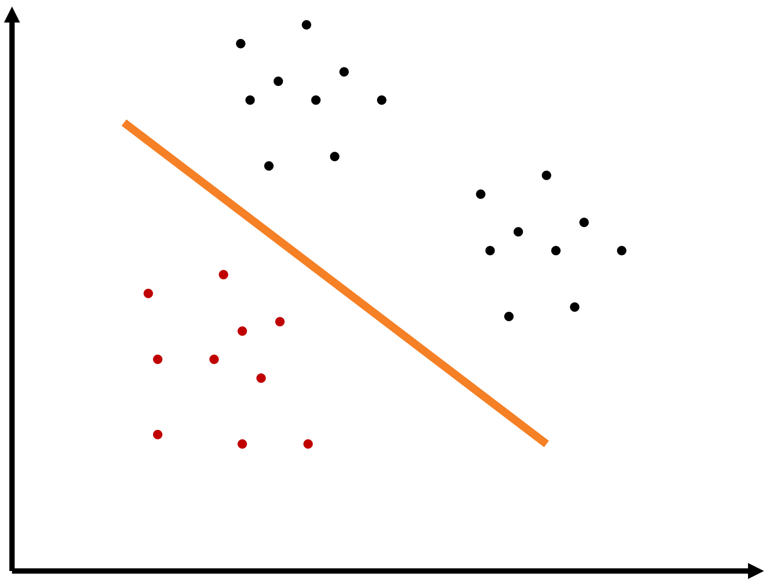
- Let's make a few concrete choices:
  - Arbitrary regions
  - Similarity based on color only
  - Similarity of regions = distance between mean colors

Divisive Clustering

- Start with whole image in one cluster
- Iterate:
  - Find cluster with largest intra-cluster variation
  - Split into two pieces that yield largest inter-cluster distance
- Stopping criteria?

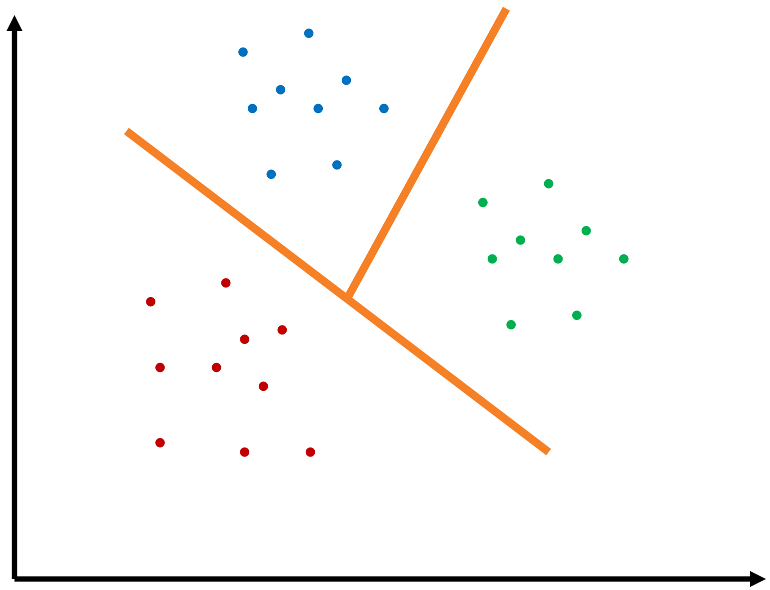
Divisive Clustering

- Start with whole image in one cluster
- Iterate:
  - Find cluster with largest intra-cluster variation
  - Split into two pieces that yield largest inter-cluster distance
- Stopping criteria?



Divisive Clustering

- Start with whole image in one cluster
- Iterate:
  - Find cluster with largest intra-cluster variation
  - Split into two pieces that yield largest inter-cluster distance
- Stopping criteria?

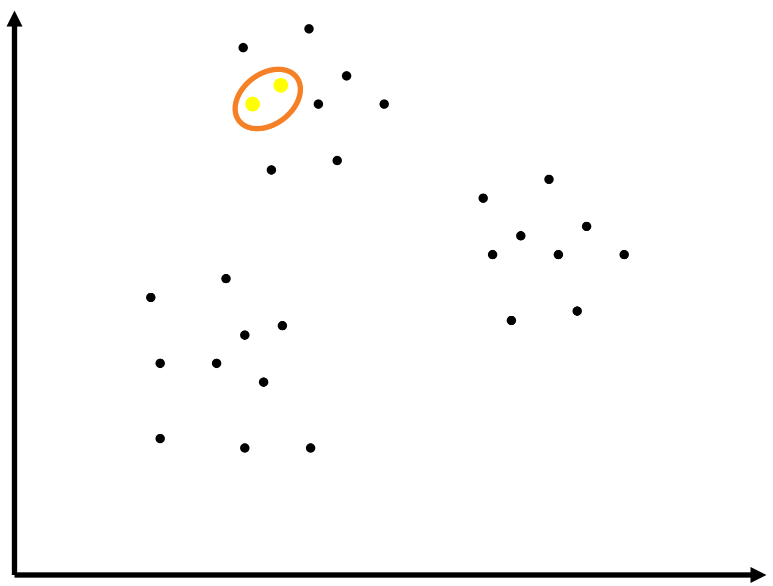


Hierarchical Clustering

- Start with each pixel in its own cluster
- Iterate:
  - Find pair of clusters with smallest inter-cluster distance
  - Merge
- Stopping criteria?

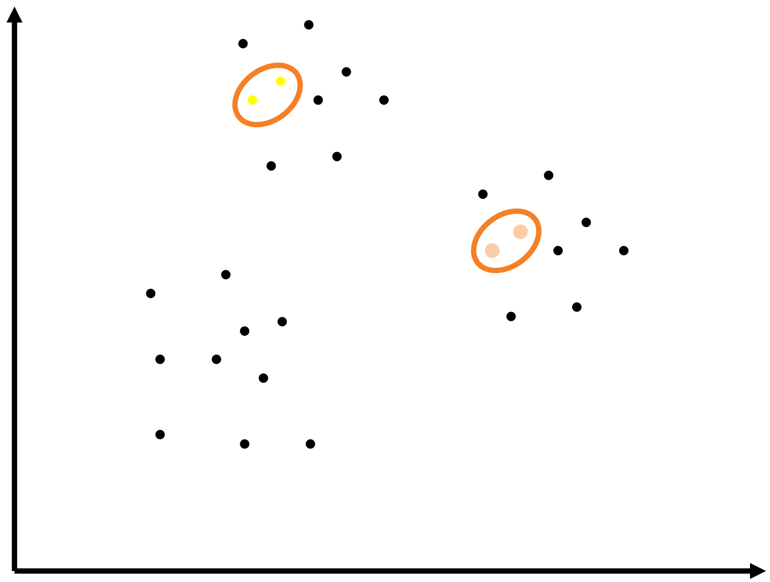
Hierarchical Clustering

- Start with each pixel in its own cluster
- Iterate:
  - Find pair of clusters with smallest inter-cluster distance
  - Merge
- Stopping criteria?



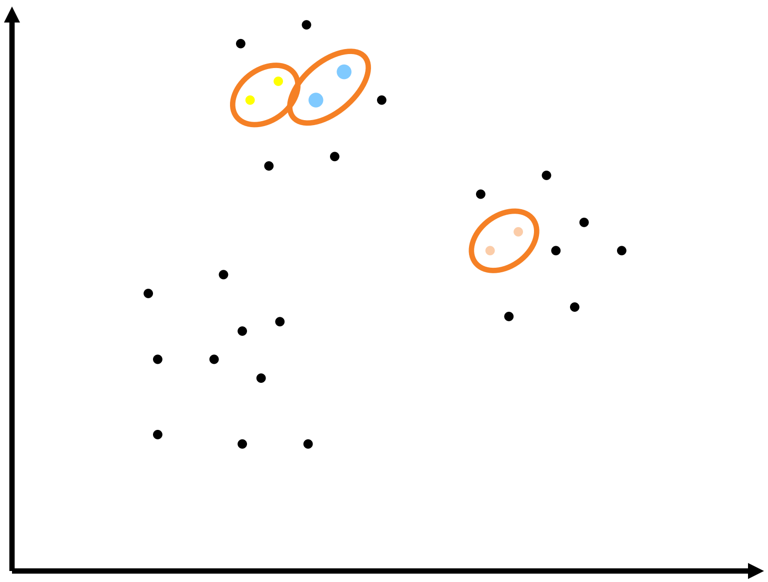
Hierarchical Clustering

- Start with each pixel in its own cluster
- Iterate:
  - Find pair of clusters with smallest inter-cluster distance
  - Merge
- Stopping criteria?



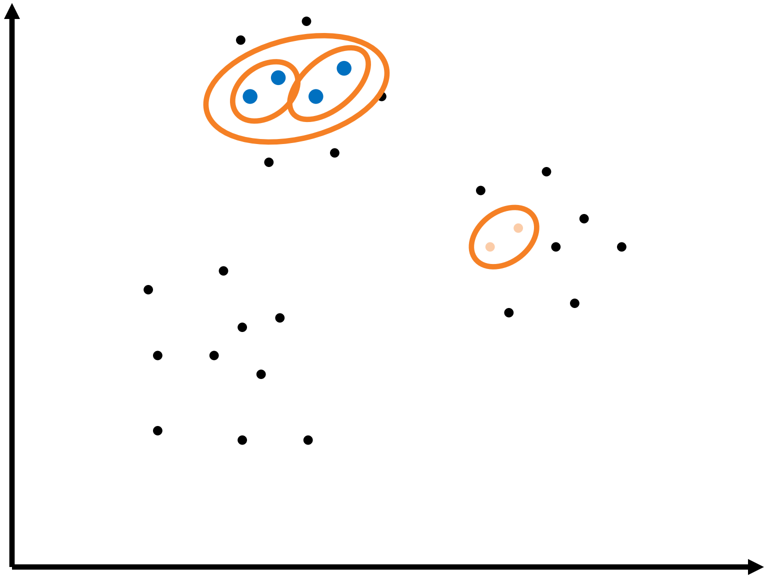
Hierarchical Clustering

- Start with each pixel in its own cluster
- Iterate:
  - Find pair of clusters with smallest inter-cluster distance
  - Merge
- Stopping criteria?



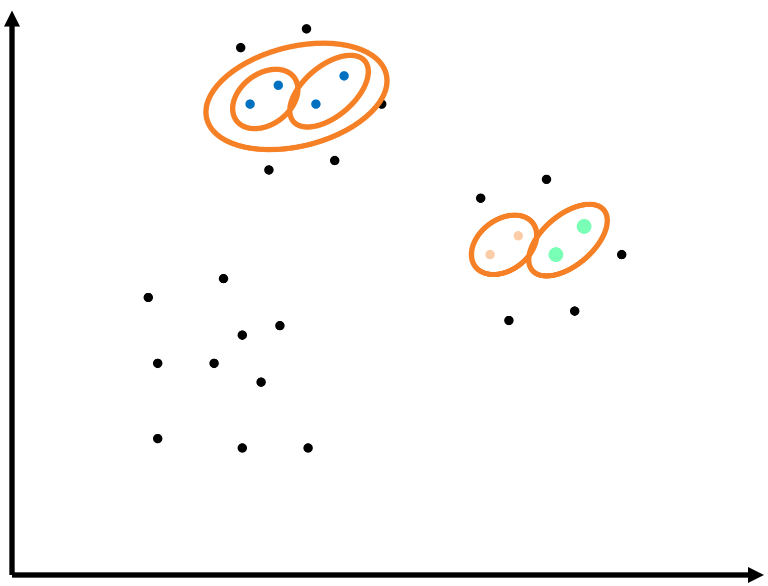
Hierarchical Clustering

- Start with each pixel in its own cluster
- Iterate:
  - Find pair of clusters with smallest inter-cluster distance
  - Merge
- Stopping criteria?



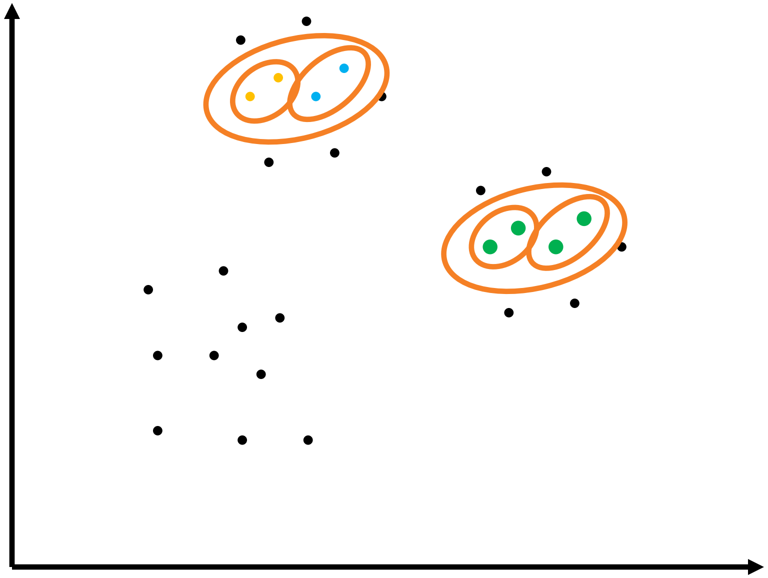
Hierarchical Clustering

- Start with each pixel in its own cluster
- Iterate:
  - Find pair of clusters with smallest inter-cluster distance
  - Merge
- Stopping criteria?



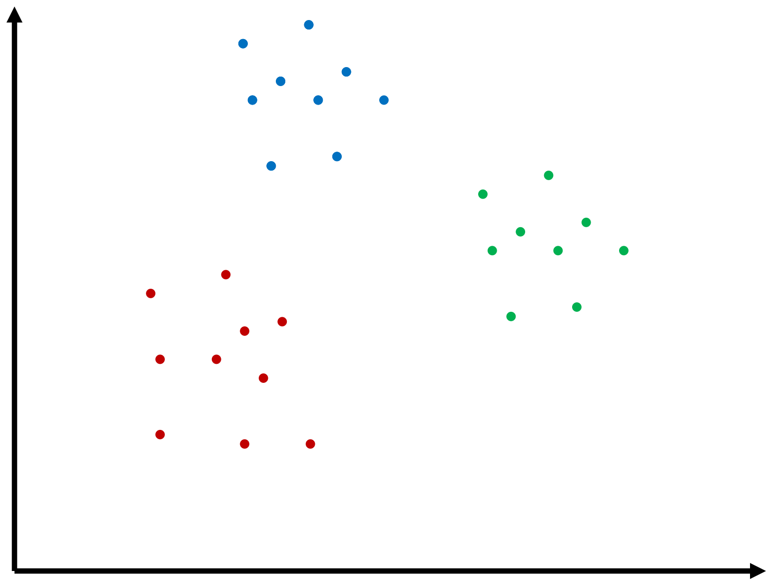
Hierarchical Clustering

- Start with each pixel in its own cluster
- Iterate:
  - Find pair of clusters with smallest inter-cluster distance
  - Merge
- Stopping criteria?



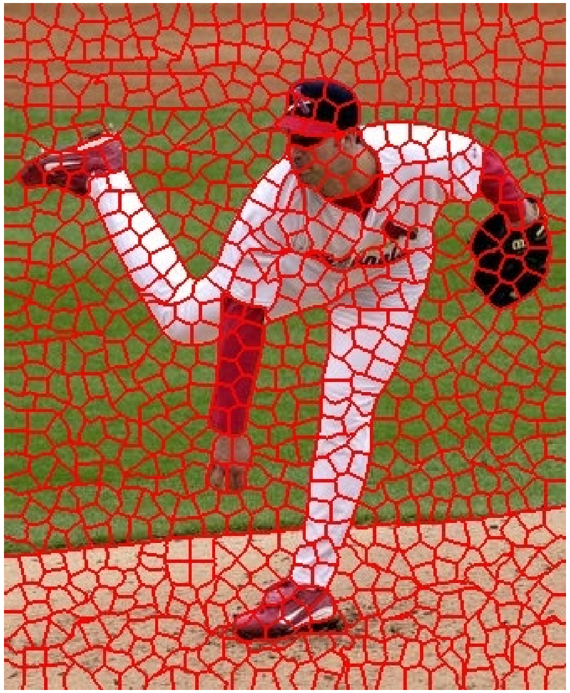
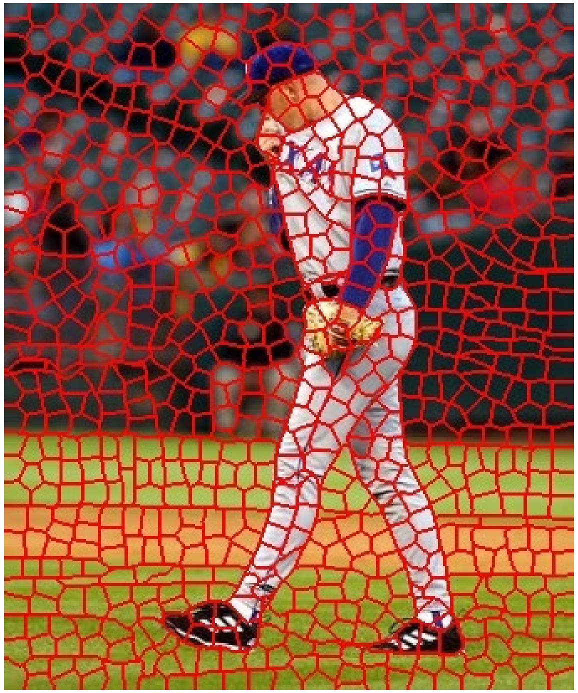
Hierarchical Clustering

- Start with each pixel in its own cluster
- Iterate:
  - Find pair of clusters with smallest inter-cluster distance
  - Merge
- Stopping criteria?



Hierarchical Clustering

- Conservative stopping criteria yields “superpixels”, which can be used as starting point for more complex algorithms

Ren et al.

Problems with These Algorithms

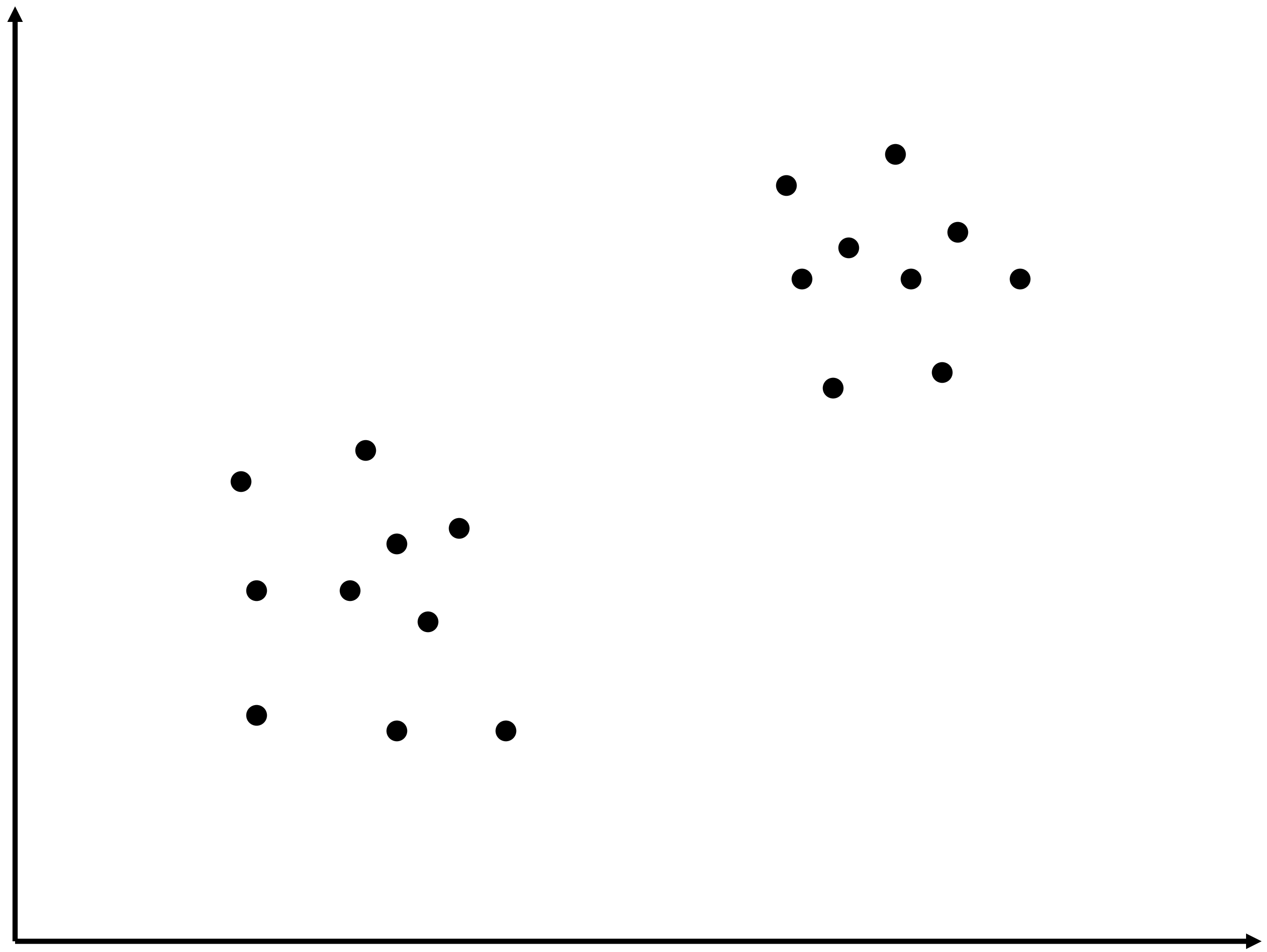
- Greedy
  - Decisions made early in process dictate final result
- Making “good” early decisions is hard/expensive
  - Many possibilities at each iteration
  - Computing “good” merge or split is expensive
- Heuristics to speed things up:
  - For agglomerative clustering, approximate each cluster by average for distance computations
  - For divisive clustering, use summary (histogram) of a region to compute split

k-means Clustering

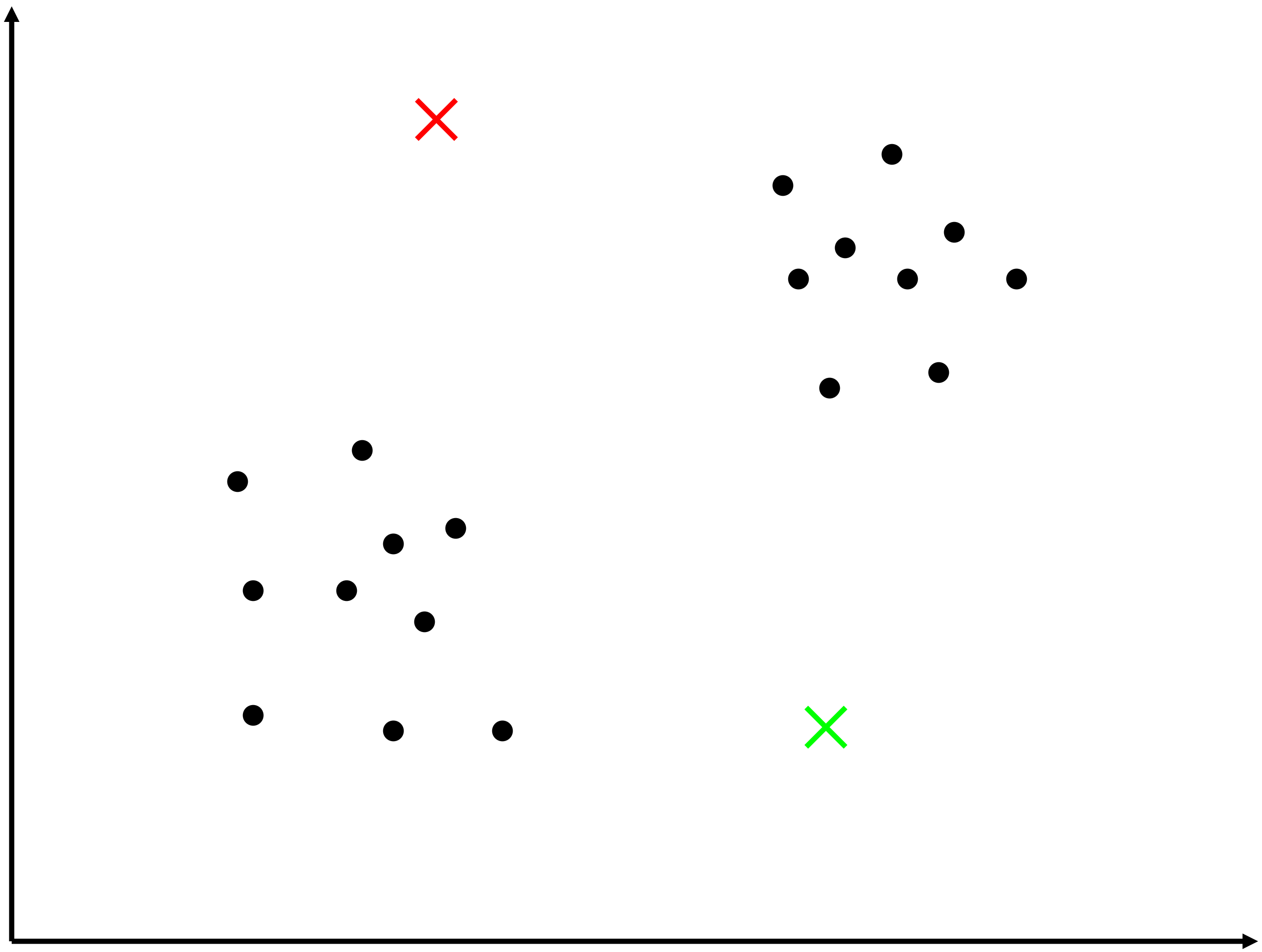
Instead of merging or splitting, start out with the clusters and move them around

1. Pick number of clusters *k*
2. Randomly scatter *k* “cluster centers” in color space
3. Repeat:
    a. Assign each data point to its closest cluster center
    b. Move each cluster center to the mean of the points assigned to it

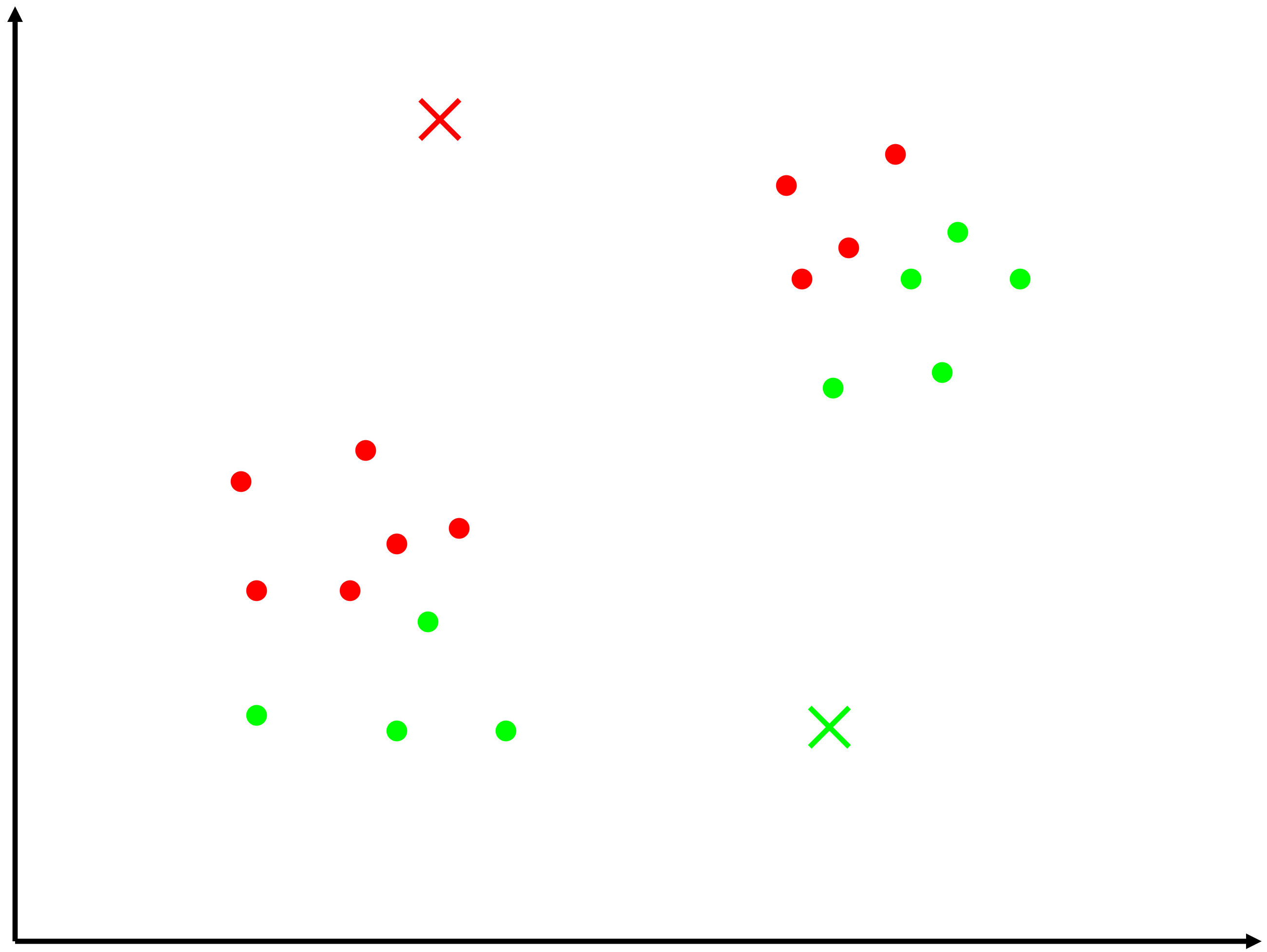
k-means Clustering



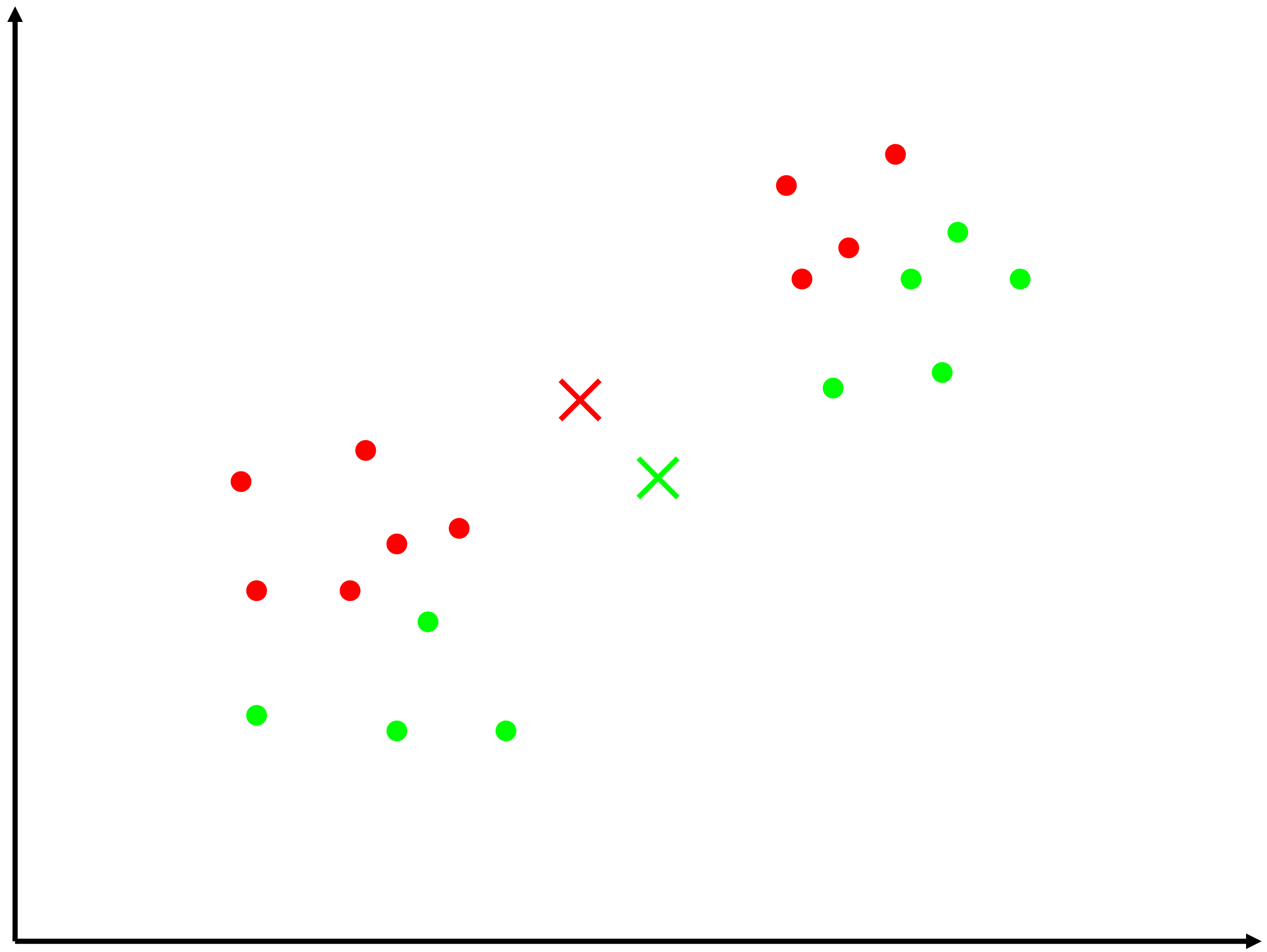
k-means Clustering



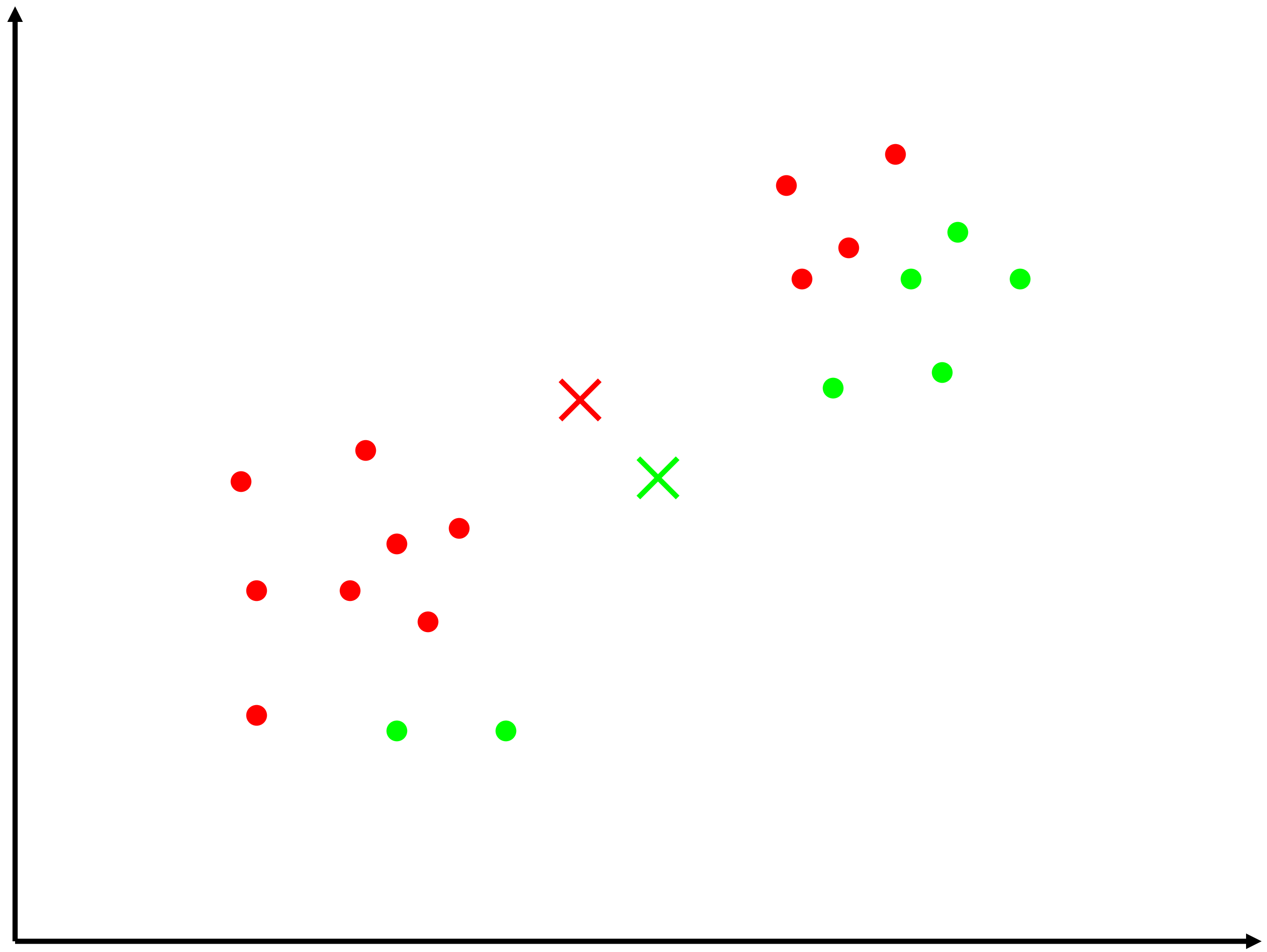
k-means Clustering



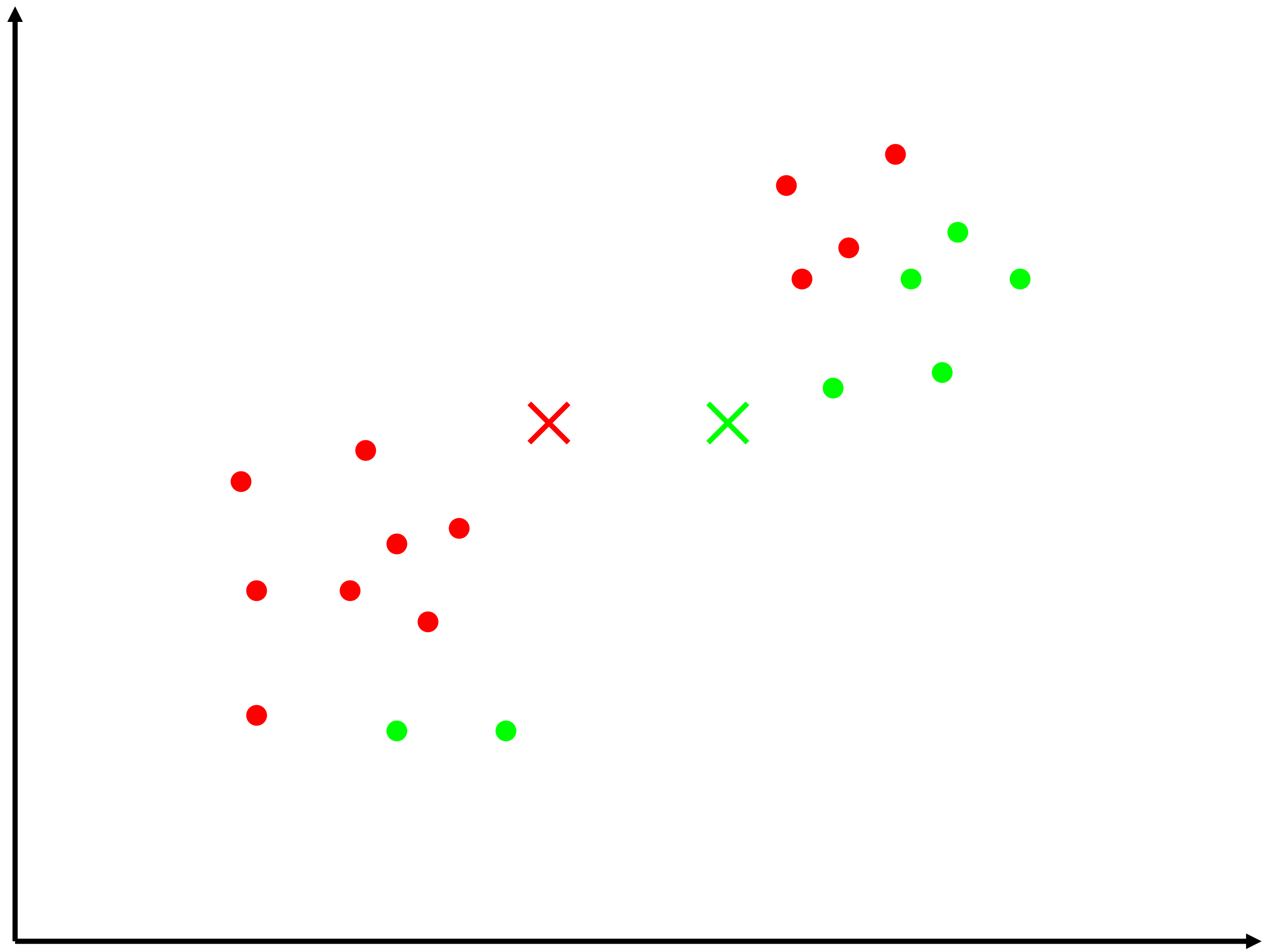
k-means Clustering



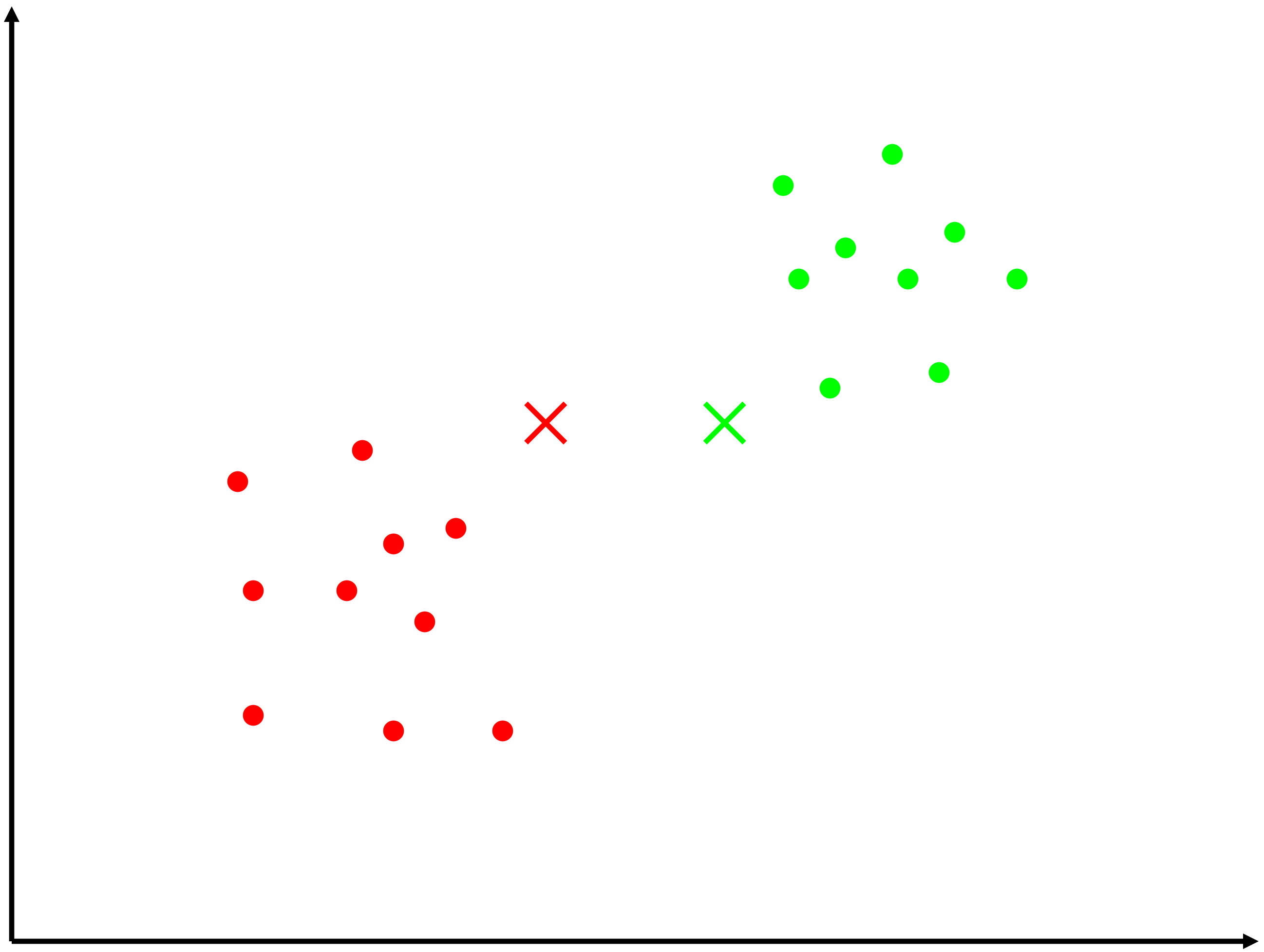
k-means Clustering



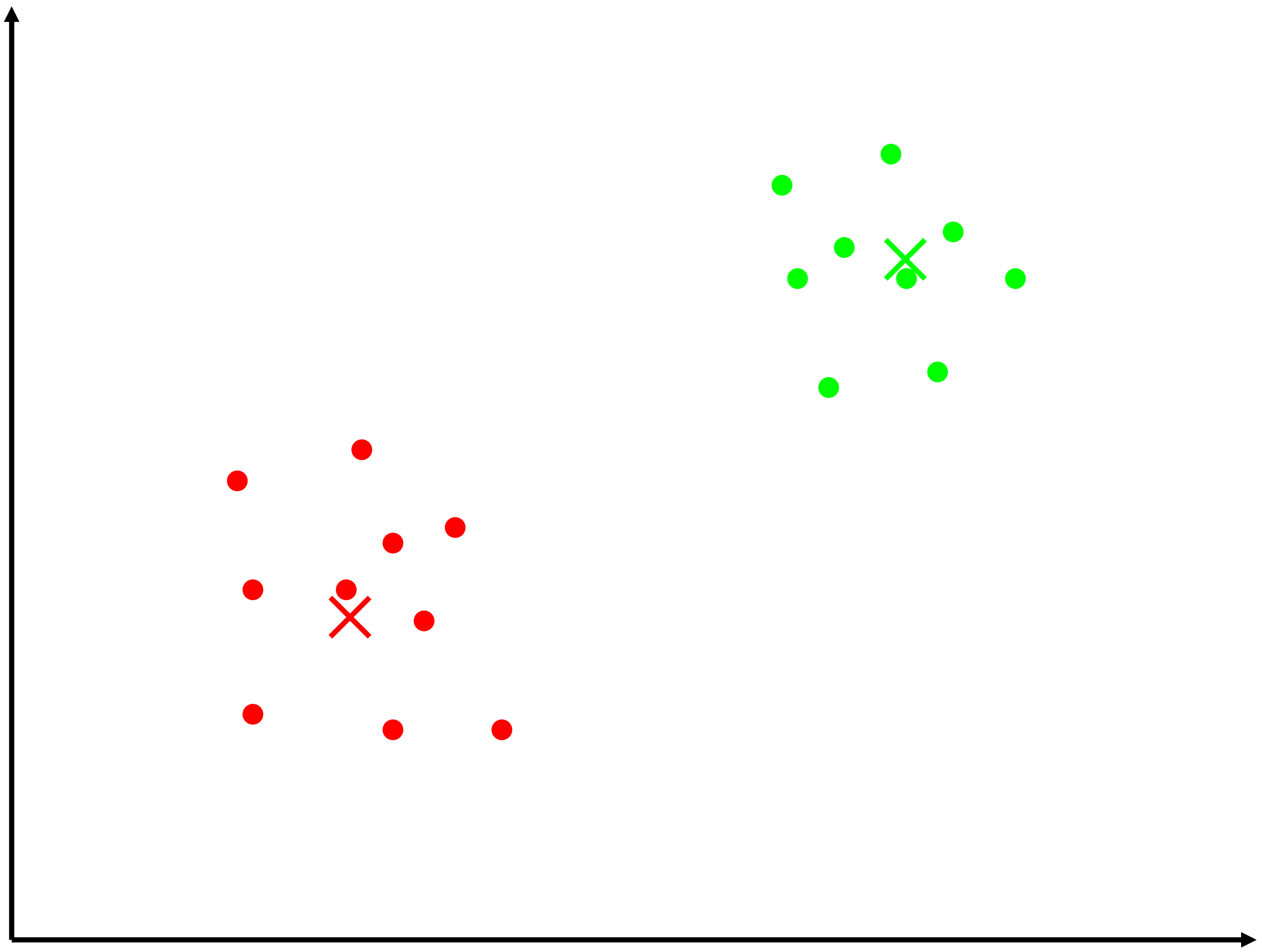
k-means Clustering



k-means Clustering



k-means Clustering



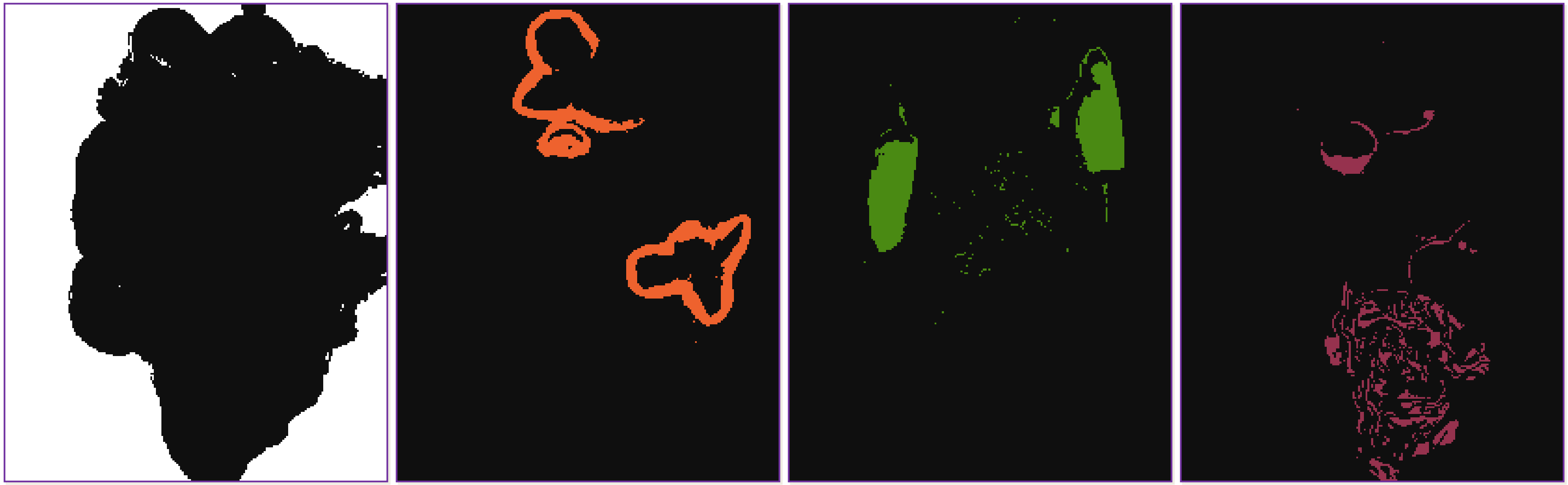
Results of Clustering

Original Image

k-means, $k=5$

k-means, $k=11$

Results of Clustering



Sample clusters with *k*-means clustering
based on color

Other Distance Measures

- Suppose we want to have compact regions
- New feature space: 5D
  (2 spatial coordinates, 3 color components)
- Points close in this space are close both in color and in actual proximity

Results of Clustering

Sample clusters with *k*-means clustering
based on color and distance

Other Distance Measures

- Problem with simple Euclidean distance: what if coordinates range from 0-1000 but colors only range from 0-255?
  - Depending on how things are scaled, gives different weight to different kinds of data
- Weighted Euclidean distance: adjust weights to emphasize different dimensions

$$\|x-y\|^2 = \sum c_i (x_i - y_i)^2$$

Mahalanobis Distance

- Automatically assign weights based on actual variation in the data

$$\left\|\vec{x}-\vec{y}\right\|^2 = \left(\vec{x}-\vec{y}\right)^{\mathrm{T}} \mathbf{C}^{-1} (\vec{x}-\vec{y})$$

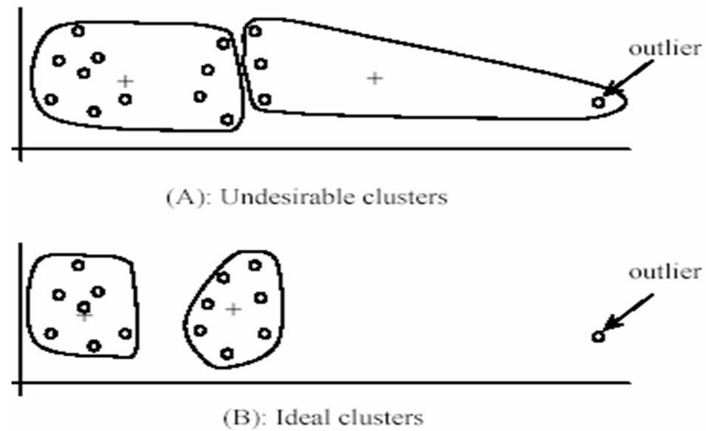
  where C is covariance matrix of all points

- Gives each dimension "equal" weight
- Also accounts for correlations between different dimensions

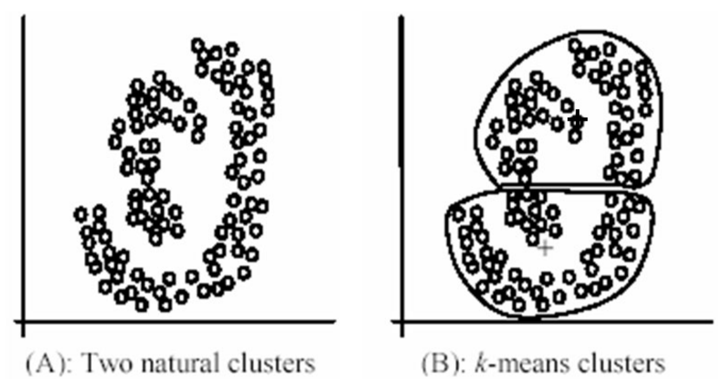
k-means Pros and Cons?

k-means Pros and Cons

- Pros
  - Very simple method
- Cons
  - Need to pick *k*
  - Converges to a local minimum
  - Sensitive to initialization
  - Sensitive to outliers
  - Only finds "spherical" clusters



Sensitive to outliers



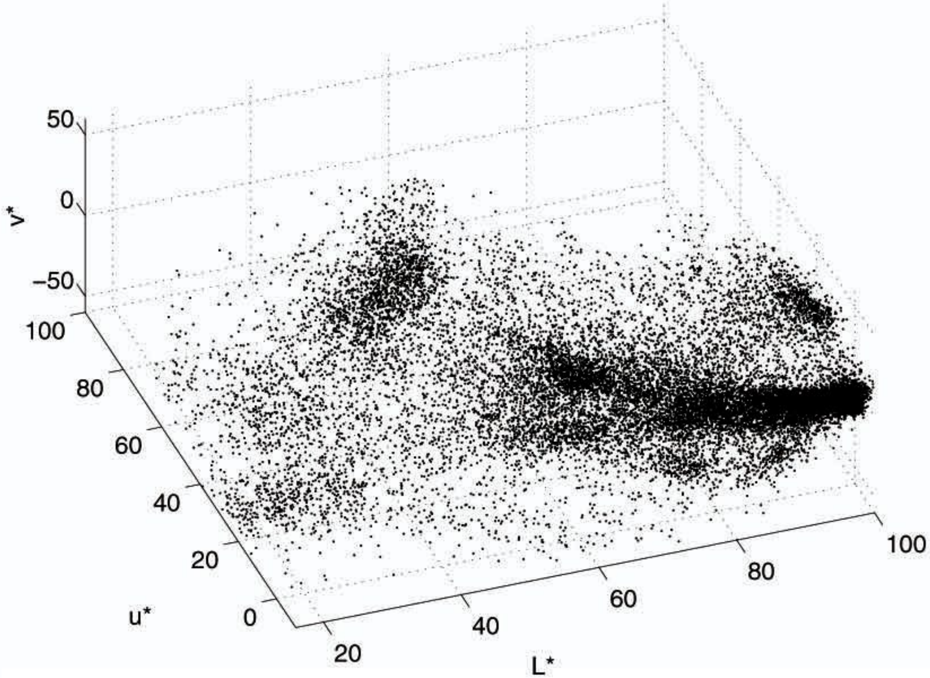
Spherical clusters

Grauman

Mean Shift Clustering

- Seek *modes* (peaks) of density in feature space

Image



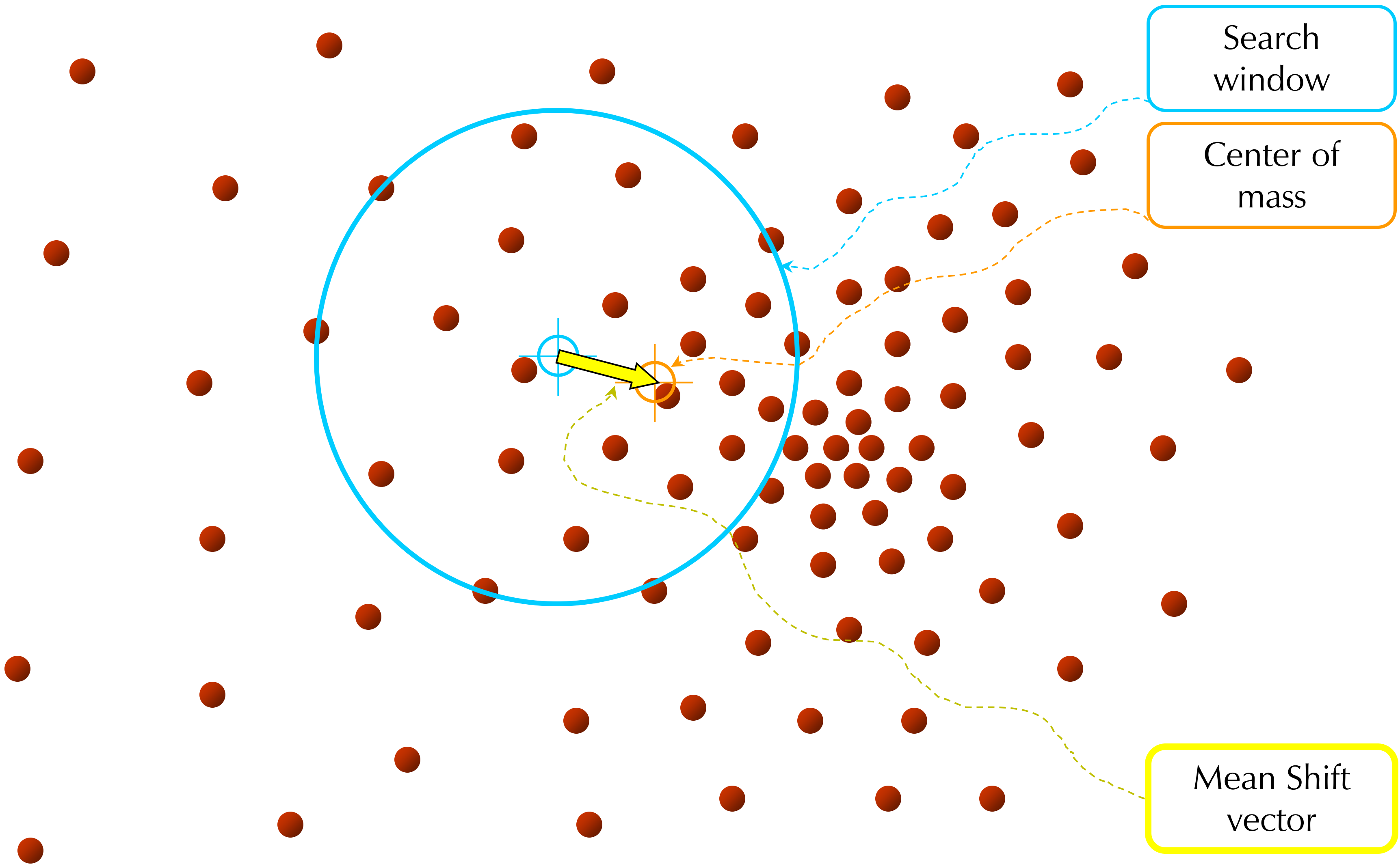
Feature space
(color values)

Ukrainitz & Sarel

Mean Shift Clustering

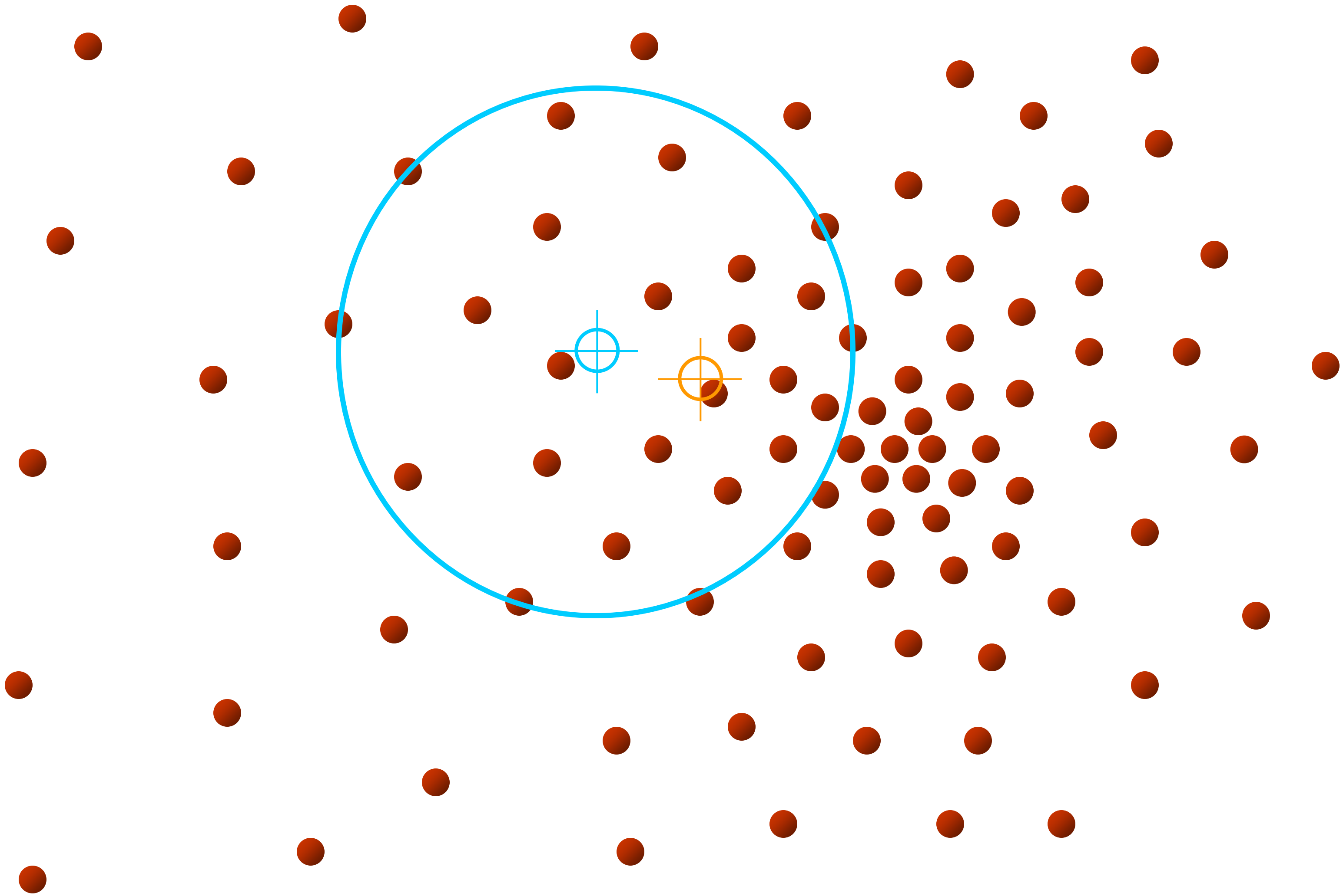
- Algorithm:
  - Initialize windows at individual feature points
  - Perform mean shift for each window until convergence
  - Merge windows that end up near the same “peak” or mode

Mean Shift Clustering



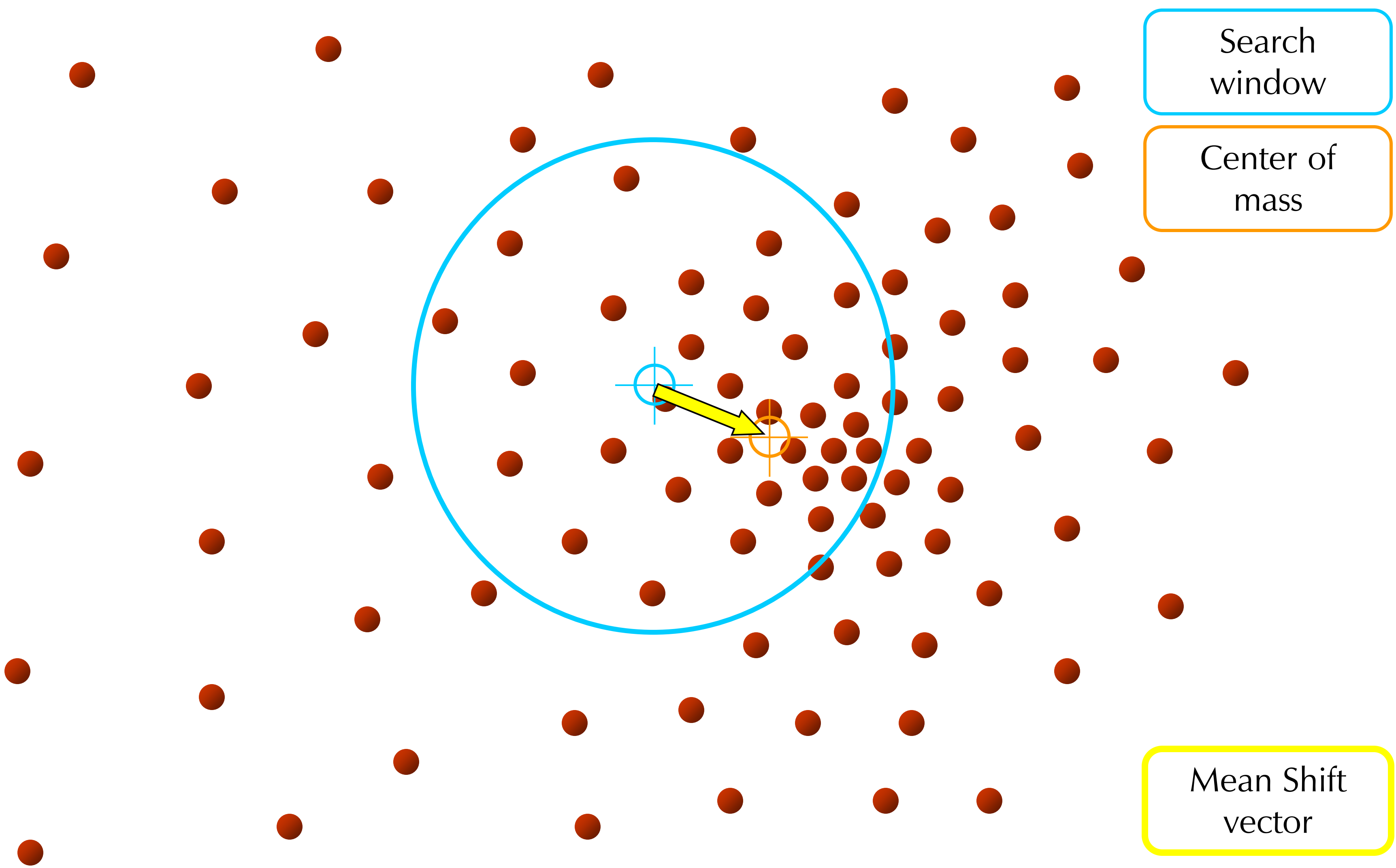
Ukrainitz & Sarel

Mean Shift Clustering



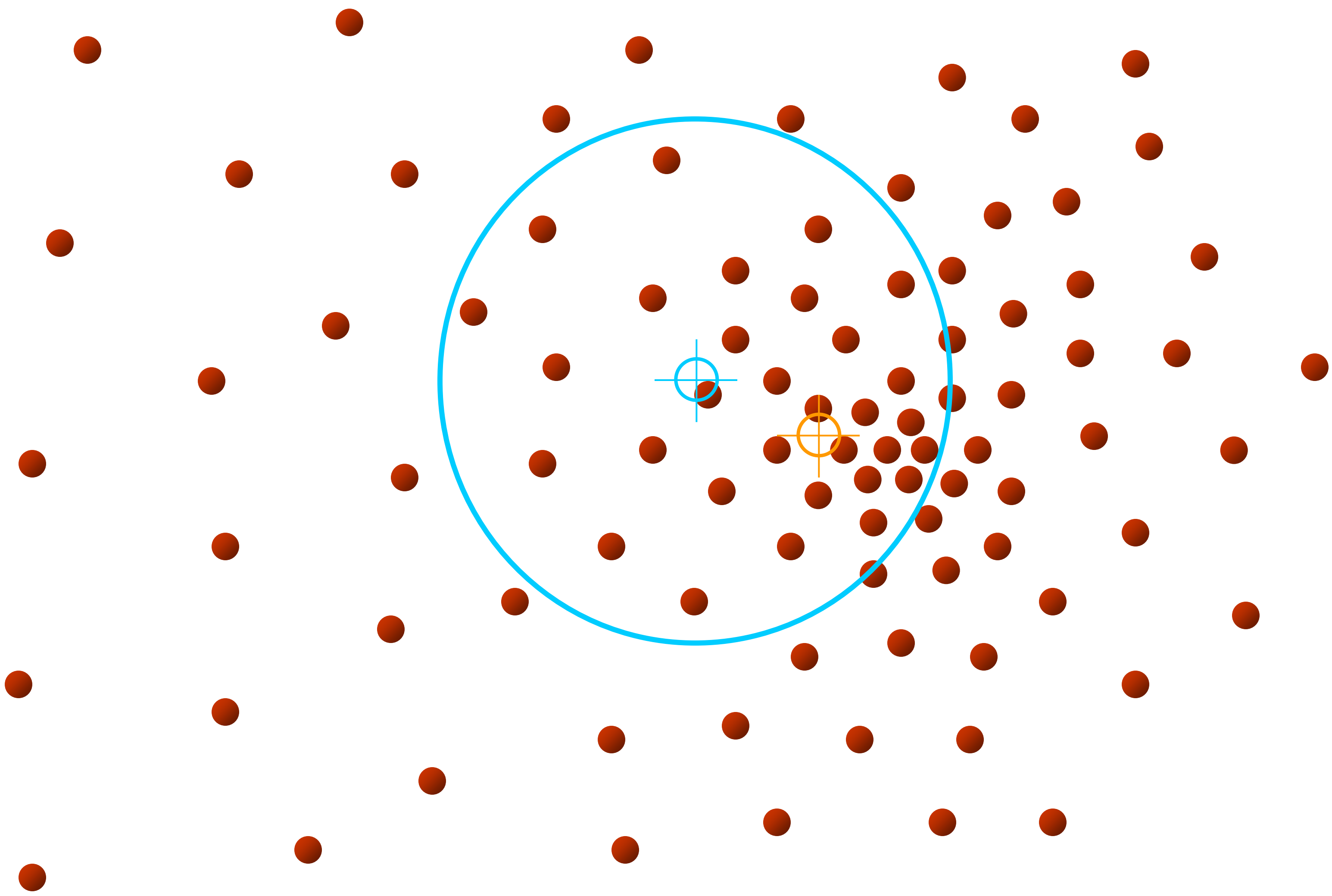
Ukrainitz & Sarel

Mean Shift Clustering



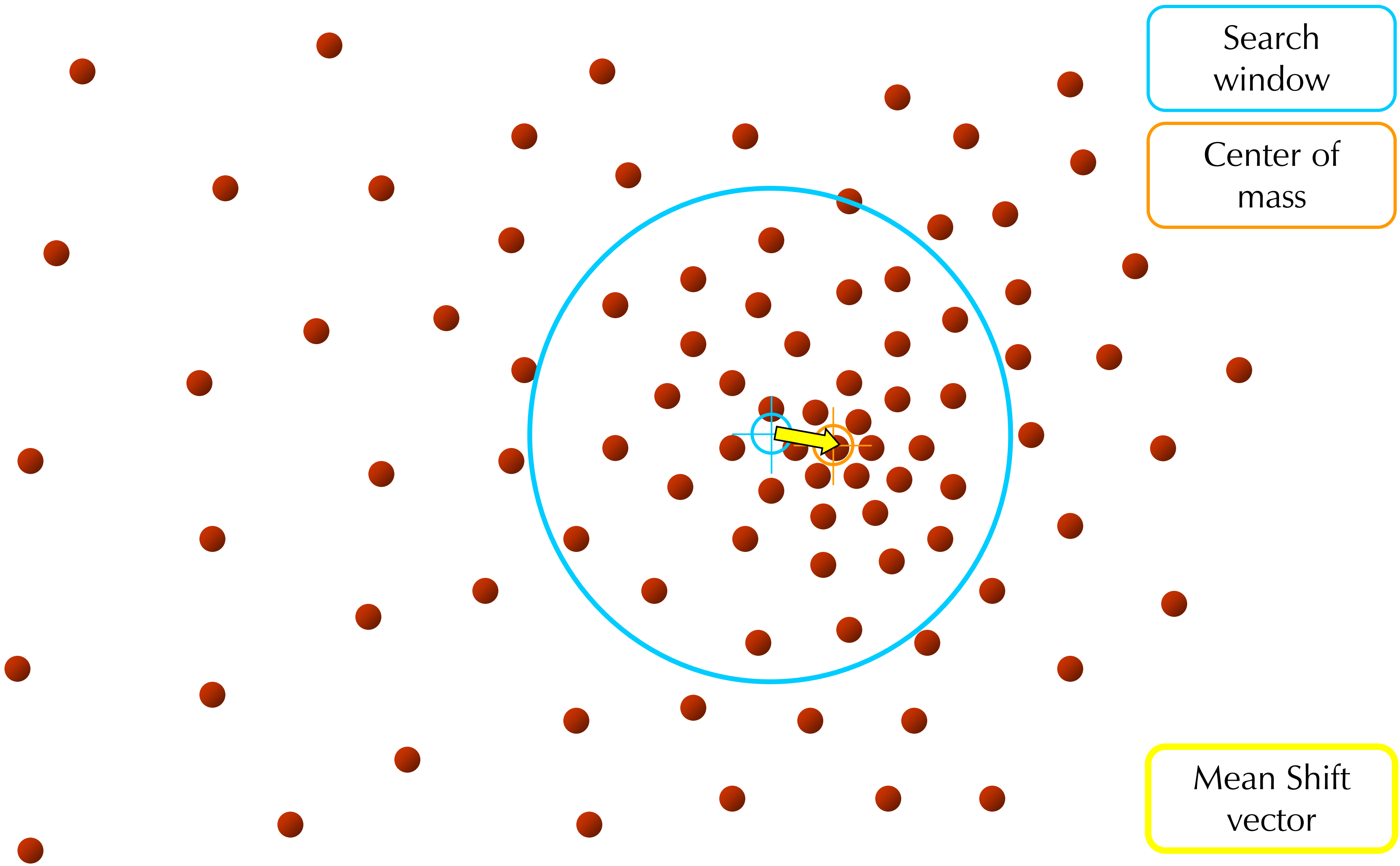
Ukrainitz & Sarel

Mean Shift Clustering



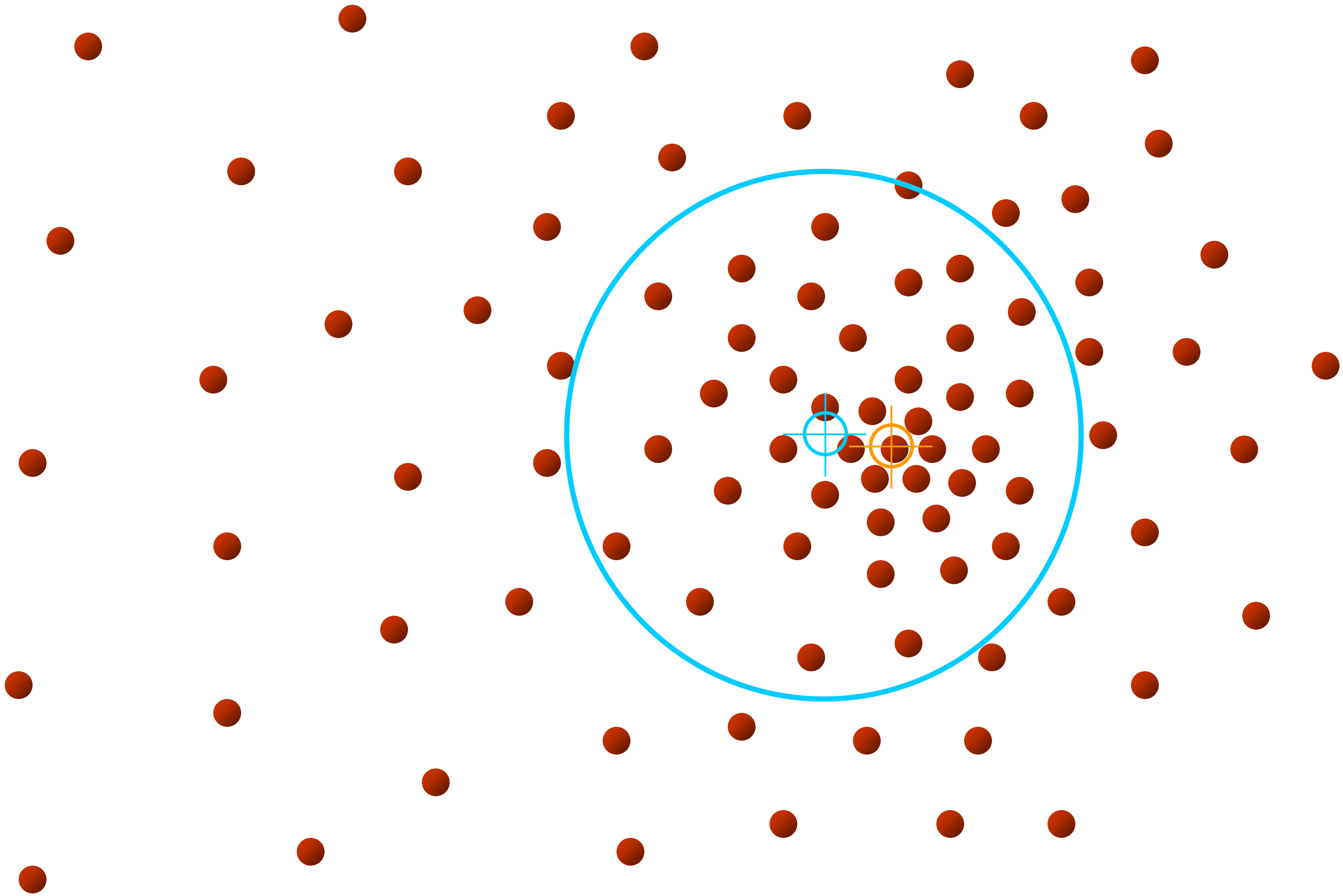
Ukrainitz & Sarel

Mean Shift Clustering



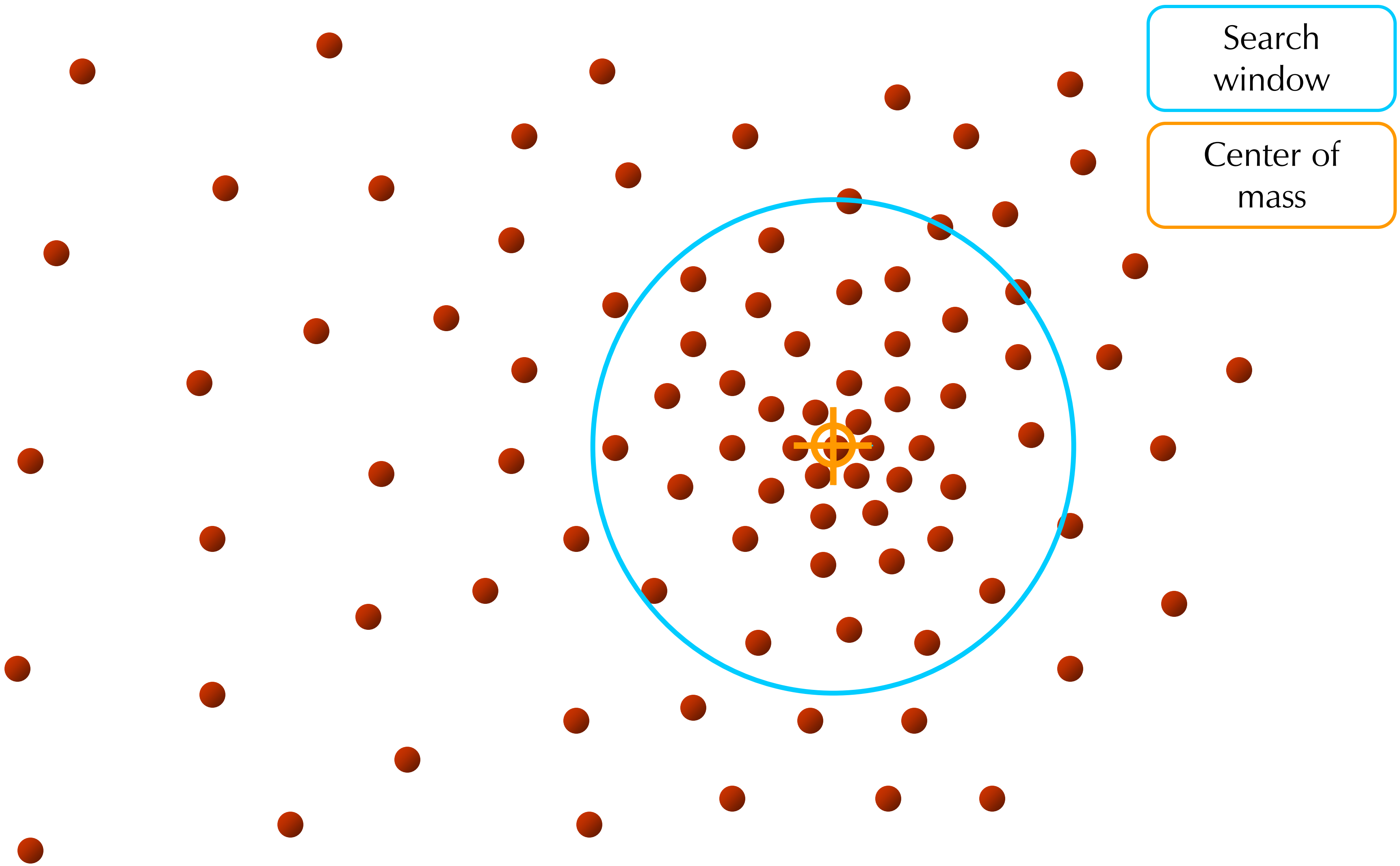
Ukrainitz & Sarel

Mean Shift Clustering



Ukrainitz & Sarel

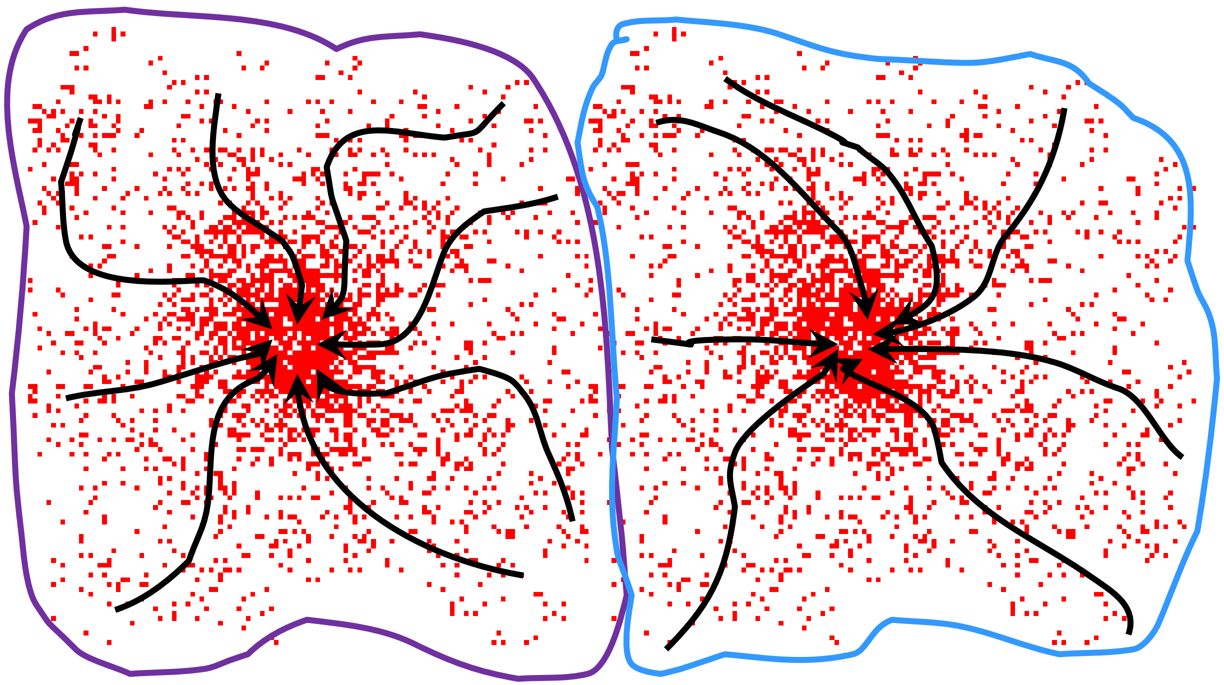
Mean Shift Clustering



Ukrainitz & Sarel

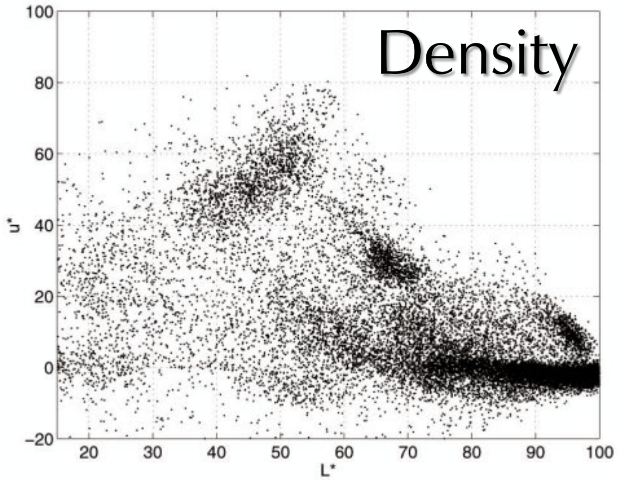
Mean Shift Clustering

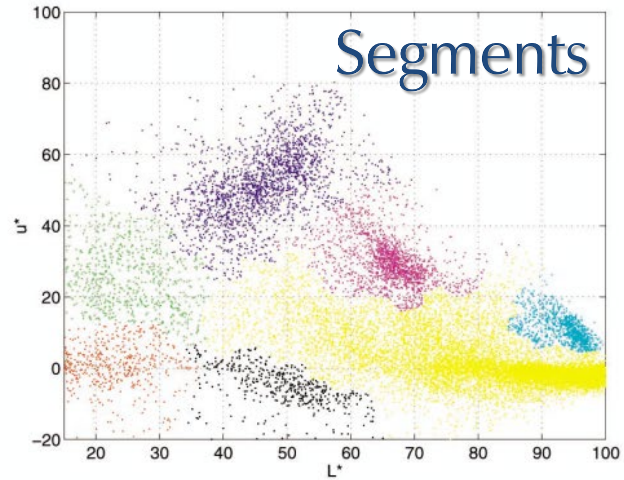
- Cluster data points in the attraction basin of a mode
  - Separate segment for each mode
  - Assign points to segments based on which mode is at the end of their mean shift trajectories

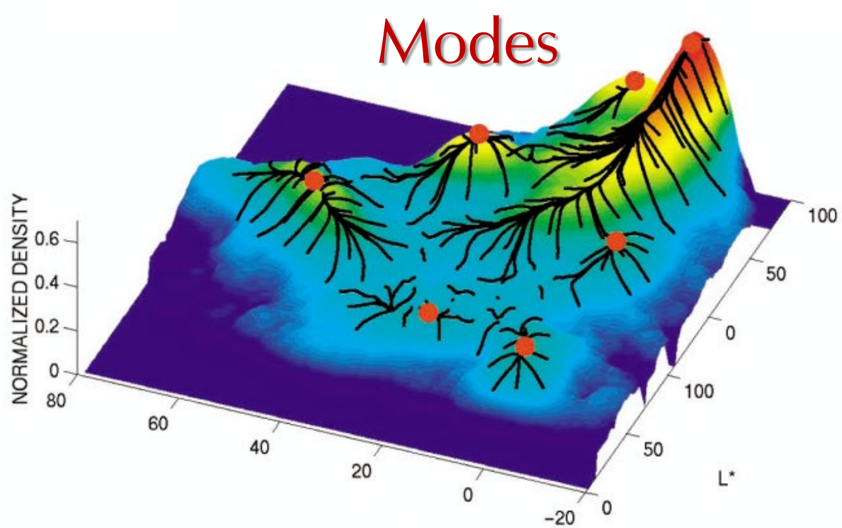


Ukrainitz & Sarel

Mean Shift Clustering

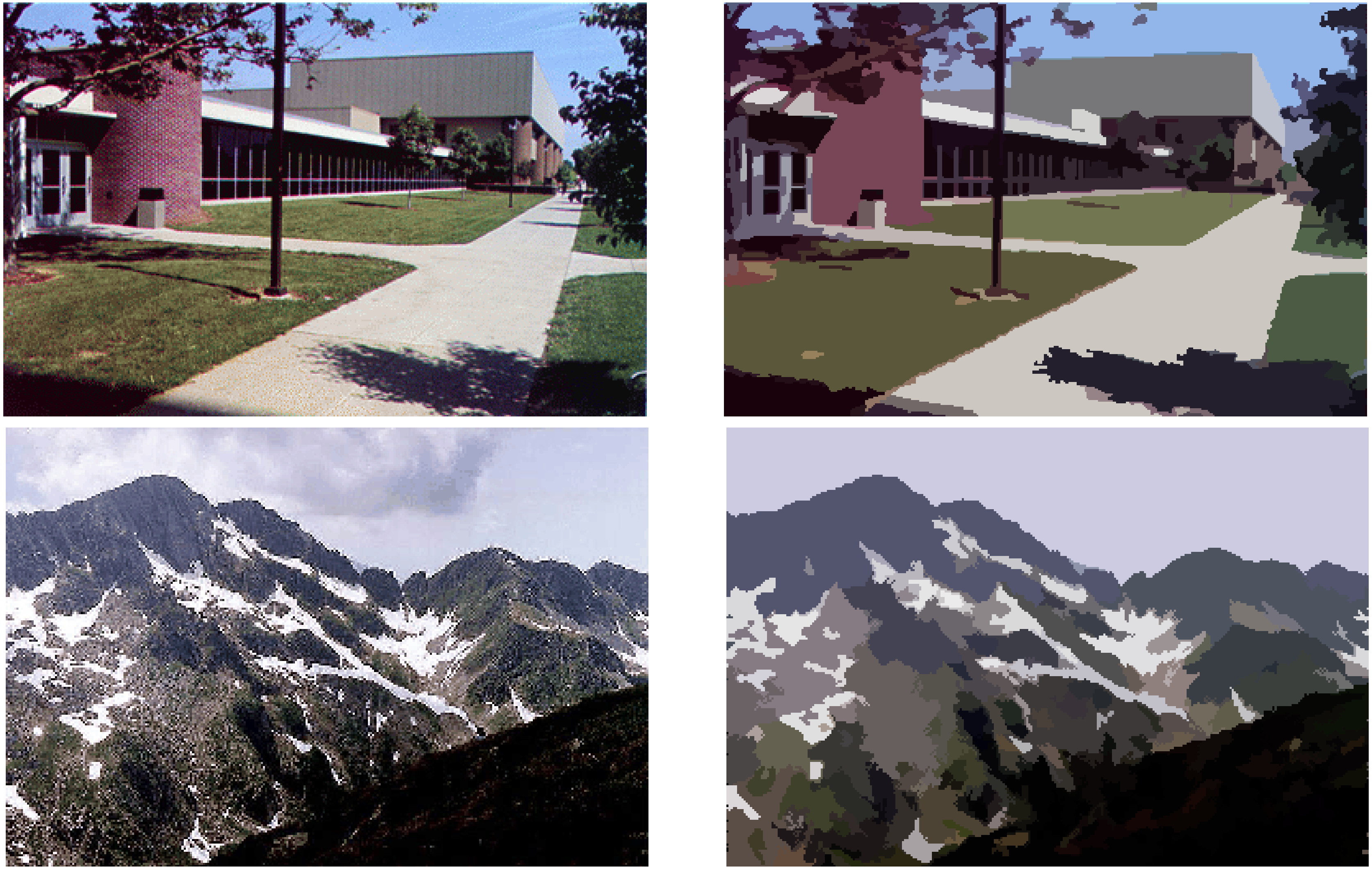






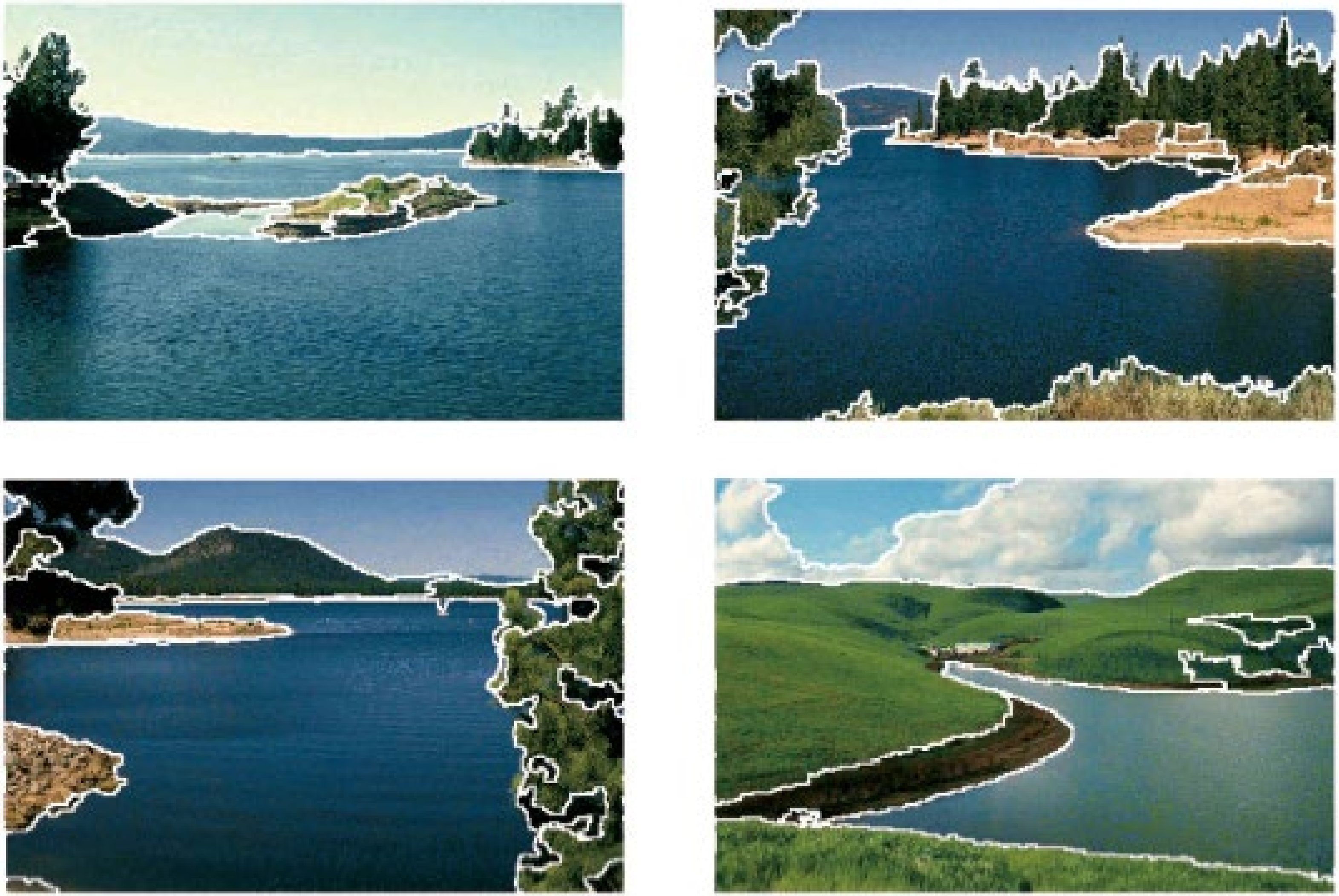
Ukrainitz & Sarel

Mean Shift Results



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Mean Shift Results



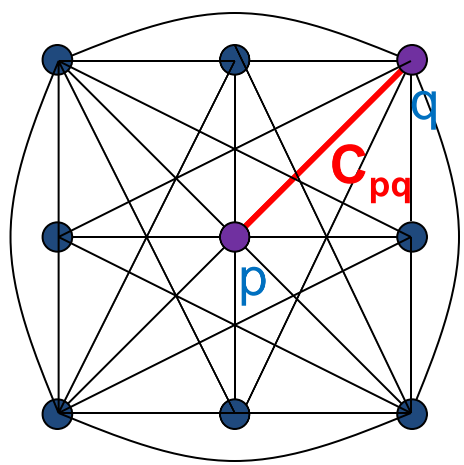
Mean Shift Pros and Cons

- Pros
  - Finds variable number of modes
  - Does not assume spherical clusters
  - Just a single parameter (window size)
  - Robust to outliers
- Cons
  - Output depends on window size
  - Computationally expensive
  - Does not scale well with dimension of feature space

Grauman

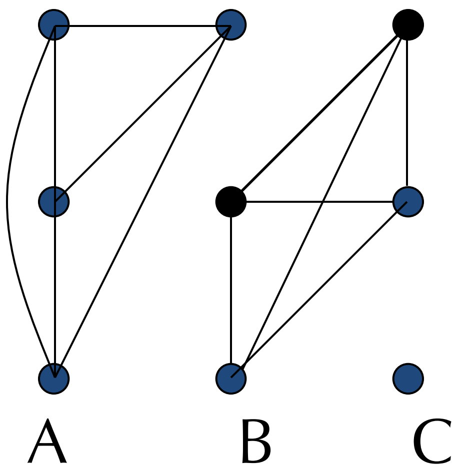
Segmentation Based on Graph Cuts

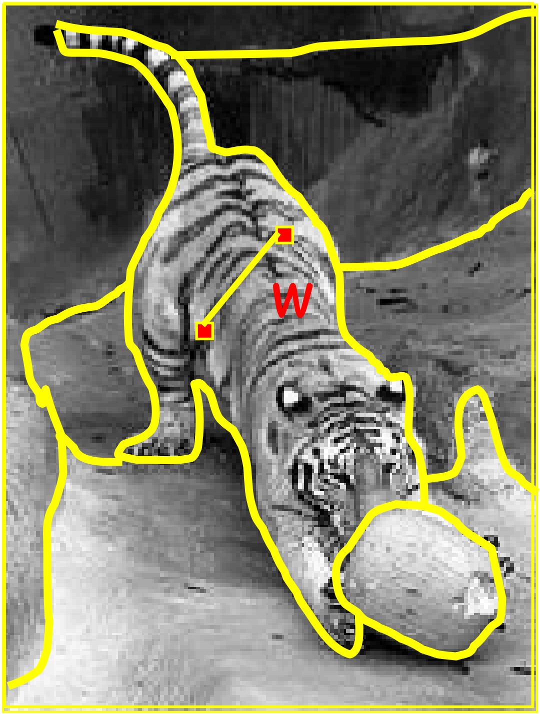
- Create weighted graph:
  - Nodes = pixels in image
  - Edge between each pair of nodes
  - Edge weight = similarity (intensity, color, texture, etc.)



Seitz

Segmentation Based on Graph Cuts

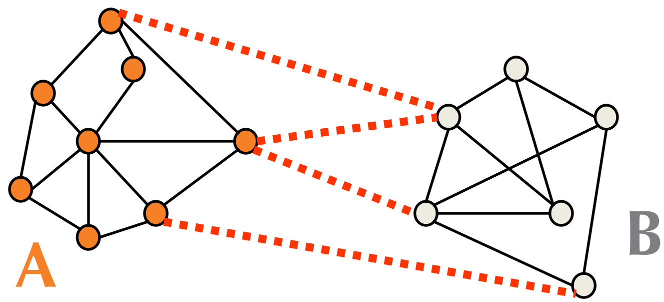




- Partition into disconnected segments
- Easiest to break links that have low cost (low similarity)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

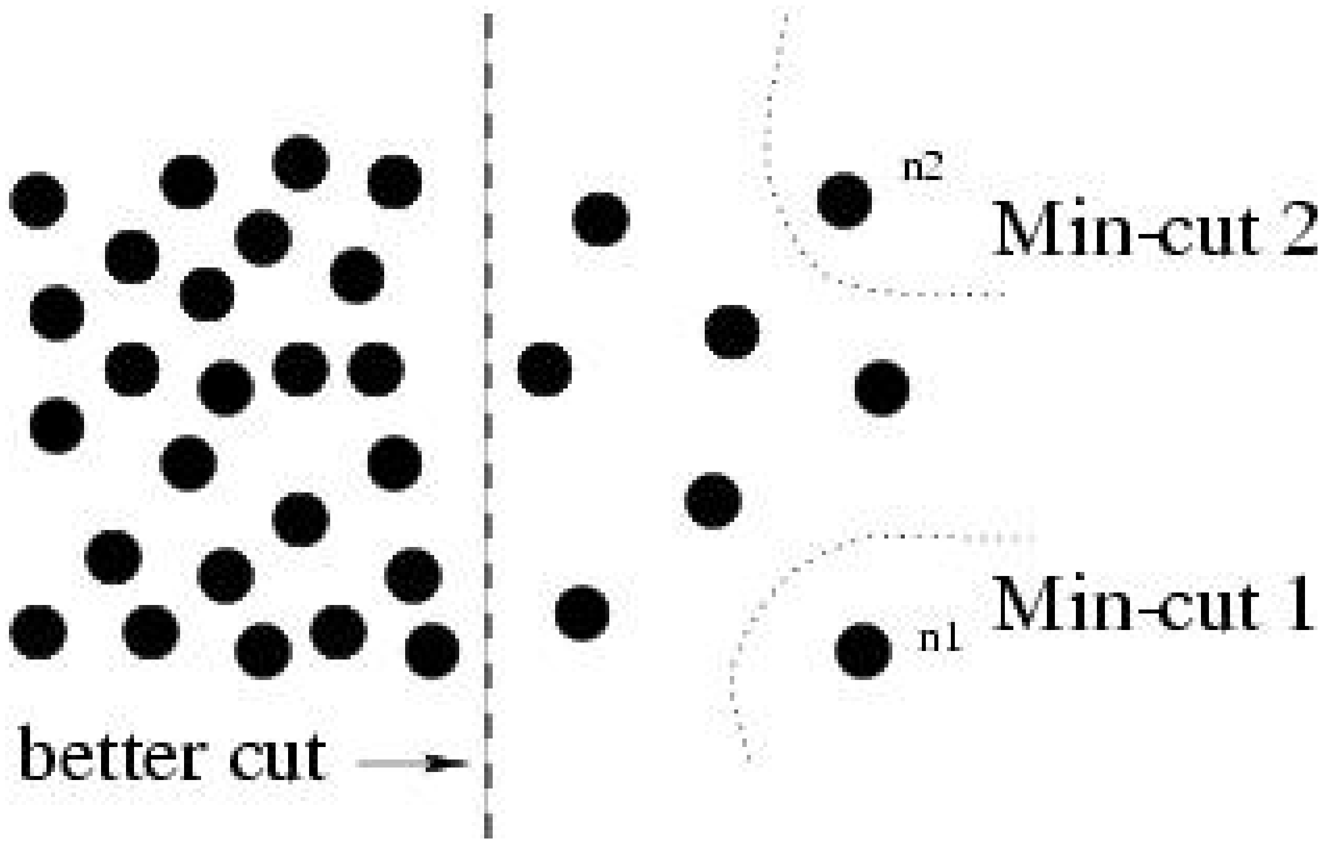
Seitz

Cuts in a Graph



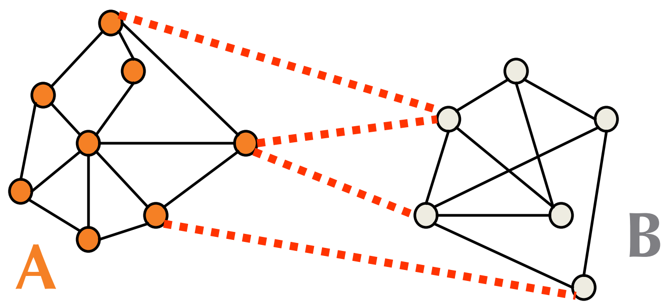
- Link Cut
  - set of links whose removal makes a graph disconnected
  - cost = sum of costs of all edges
- Min-cut
  - fast (polynomial-time) algorithm
  - gives segmentation

But Min Cut Is Not Always the Best Cut...



Seitz

Cuts in a Graph

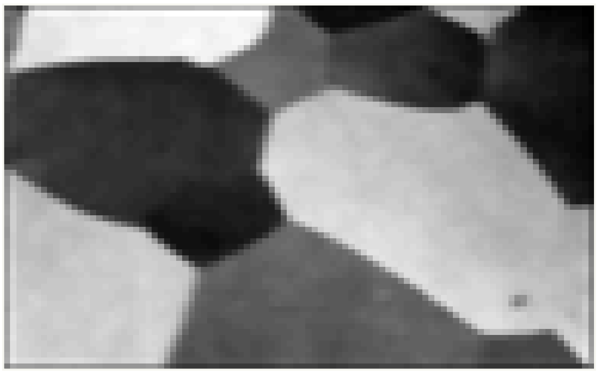


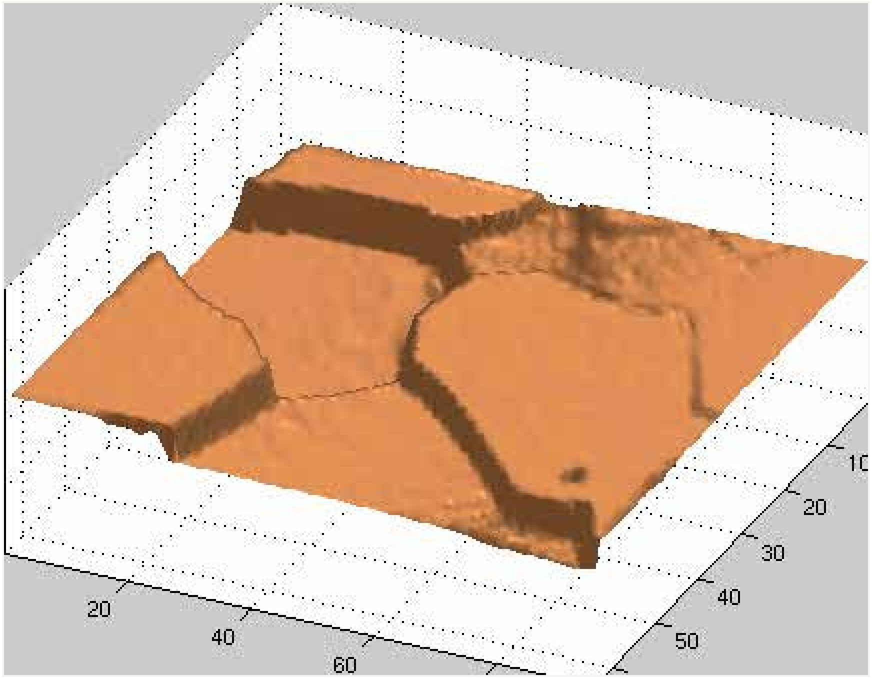
- Normalized Cut
  - removes penalty for large segments

$$Ncut(A,B) = \frac{cut(A,B)}{volume(A)} + \frac{cut(A,B)}{volume(B)}$$

  - volume(A) = sum of costs of all edges that touch A
  - no fast exact algorithms…

Interpretation as a Dynamical System





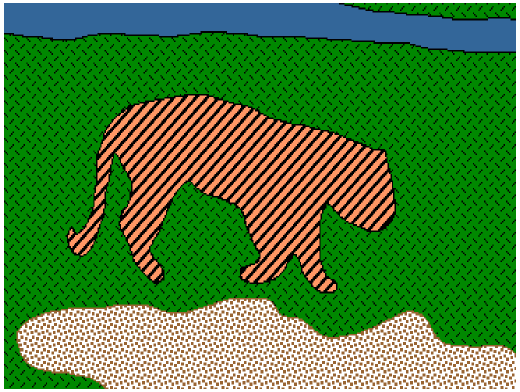
Treat the links as springs and shake the system

- elasticity proportional to cost
- vibration "modes" correspond to segments
  - can compute these by solving a generalized eigenvector problem
  - for more details, see

J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR, 1997

Seitz

Designing Grouping Features



Low-level cues

- Brightness similarity
- Color similarity
- Texture similarity

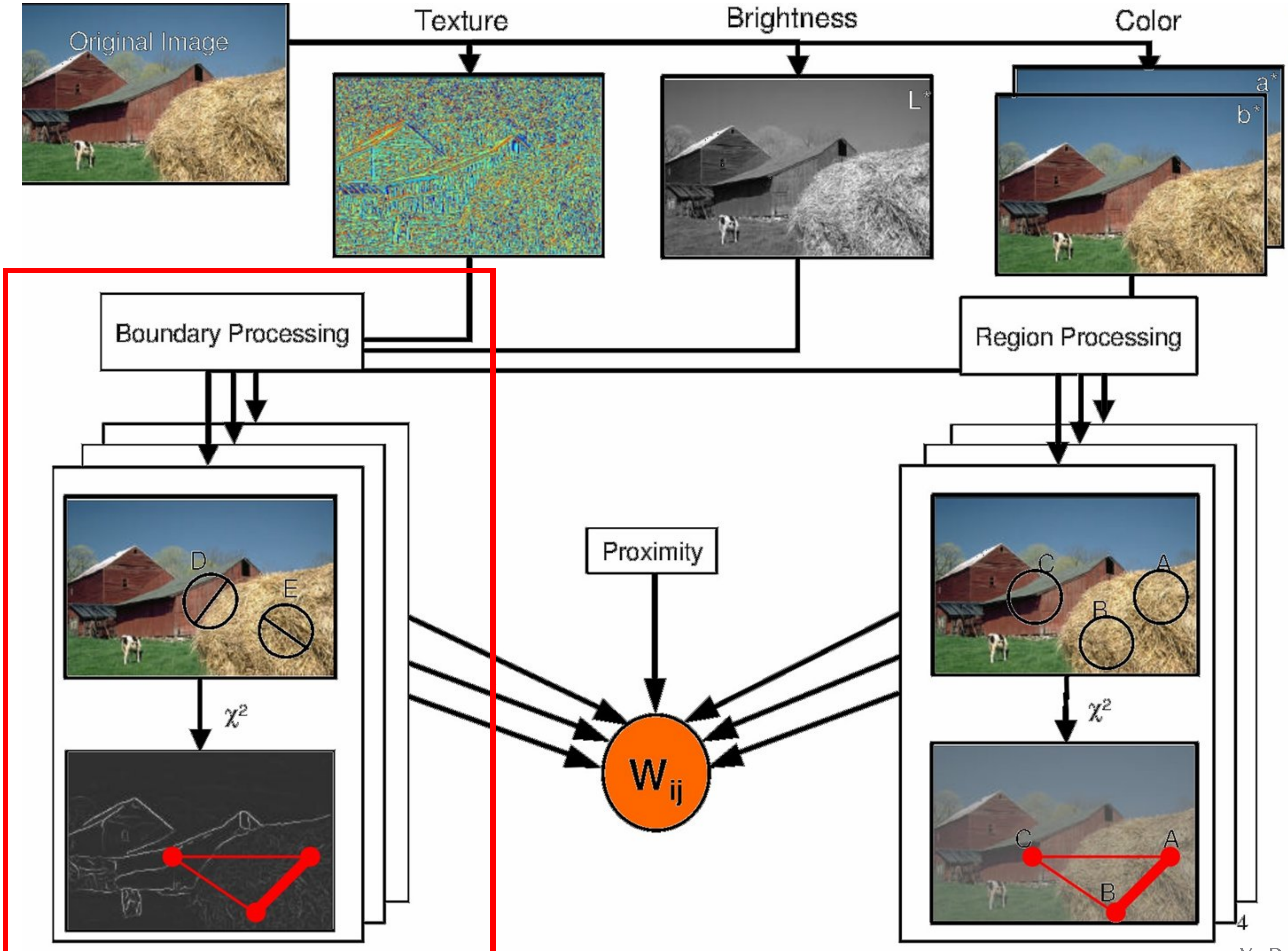
Mid-level cues

- Contour continuity
- Convexity
- Parallelism
- Symmetry

High-level cues

- Object knowledge
- Scene structure

X. Ren


Original Image
Texture
Brightness
Color
L*
a*
b*
Boundary Processing
Region Processing
D
E
C
A
B
χ^2
χ^2
Proximity
W_{ij}
C
A
B

X. Ren

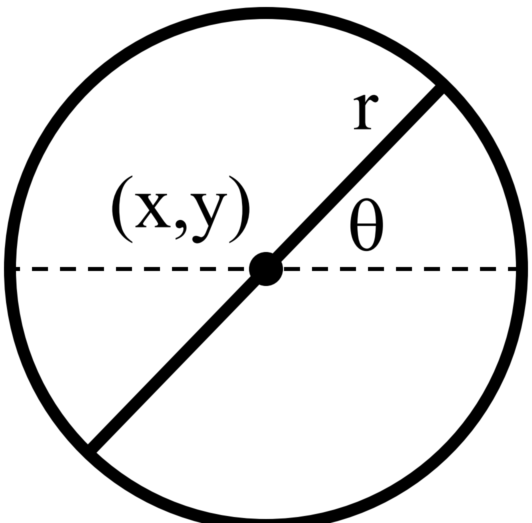
Brightness and Color Contrast

- Color (e.g., 1976 CIE L*a*b* colorspace)
- Brightness Gradient BG(x,y,r,θ)

  χ^2 difference in L* distribution
- Color Gradient CG(x,y,r,θ)

  χ^2 difference in a* and b* distributions

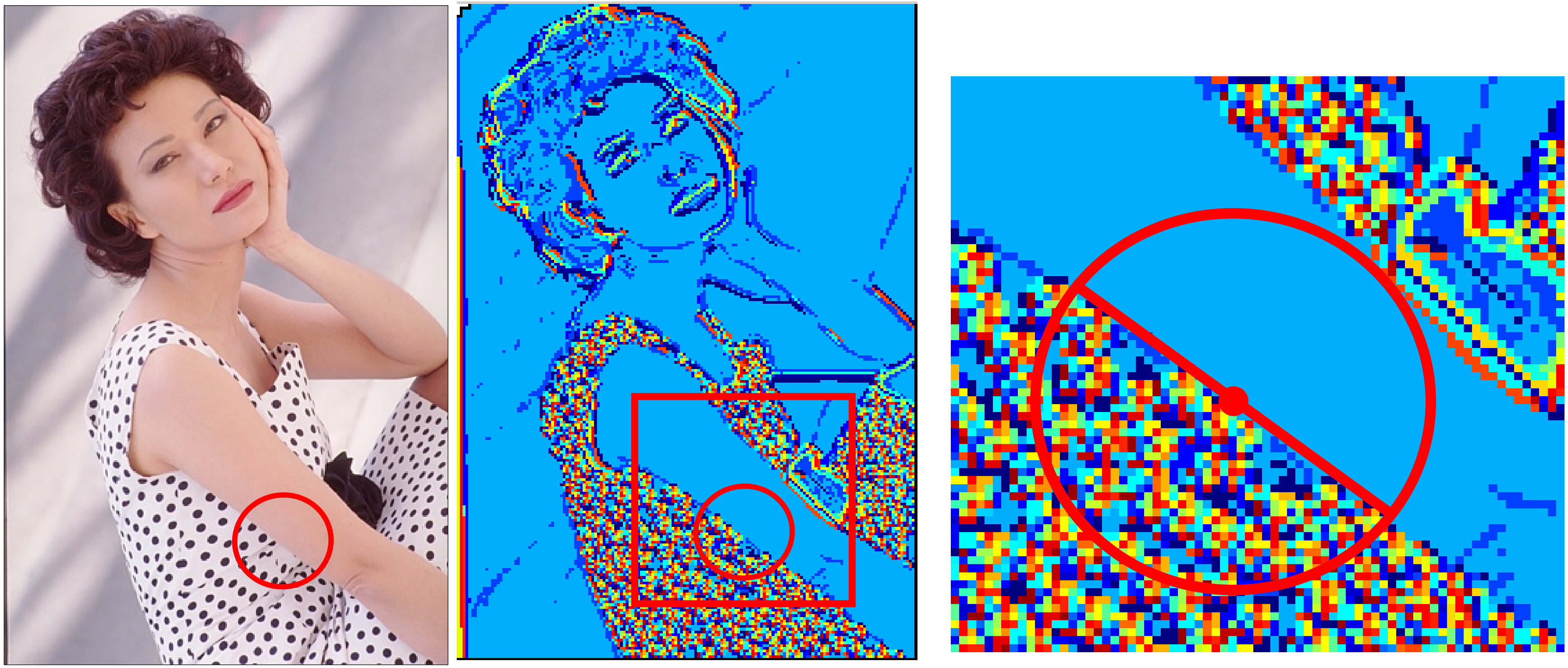
$$\chi^2(g,h) = \frac{1}{2}\sum_i \frac{(g_i - h_i)^2}{g_i + h_i}$$



X. Ren

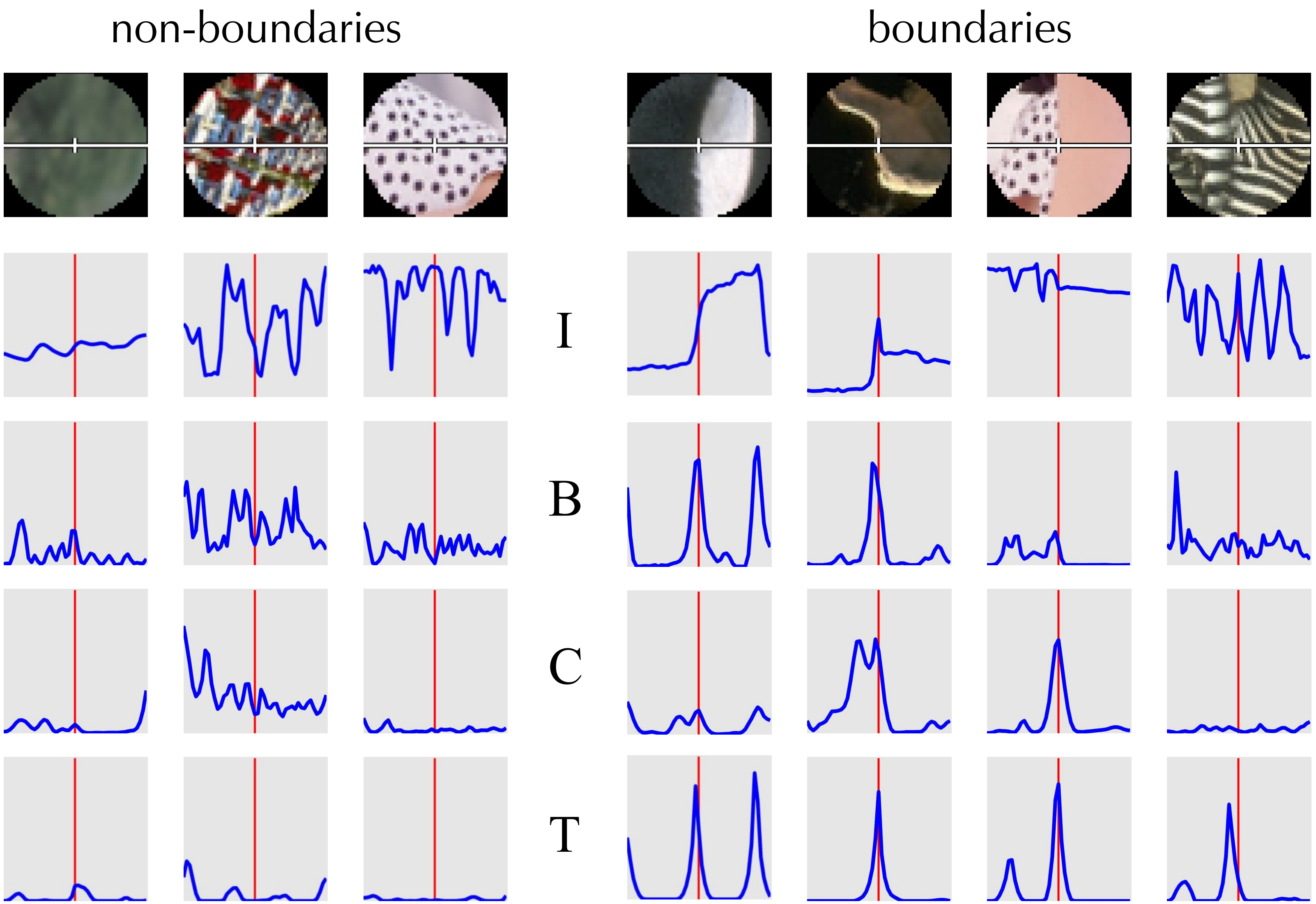
Texture Contrast

- Texture Gradient TG(x,y,r,θ)
  - χ^2 difference of texton histograms
  - Textons are vector-quantized filter outputs (through k-means)



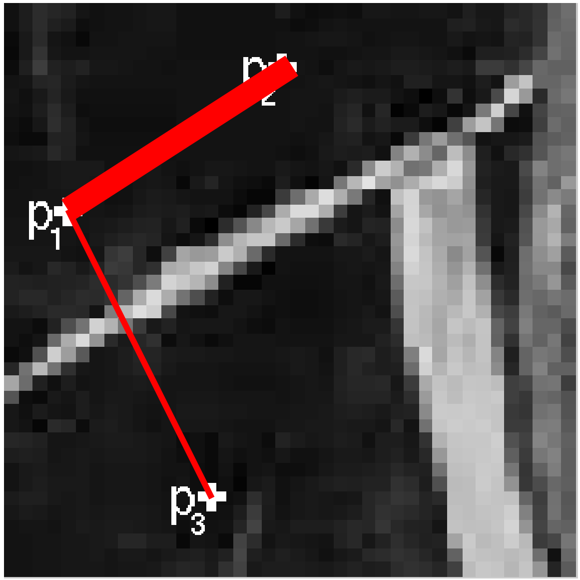
X. Ren

Boundary Classification



X. Ren

Affinity using Intervening Contour



W(p1,p2) >>W(p1,p3) as p1 and p2 are more likely to belong to the same region than are p1 and p3, which are separated by a strong boundary.

X. Ren

Combining Cues

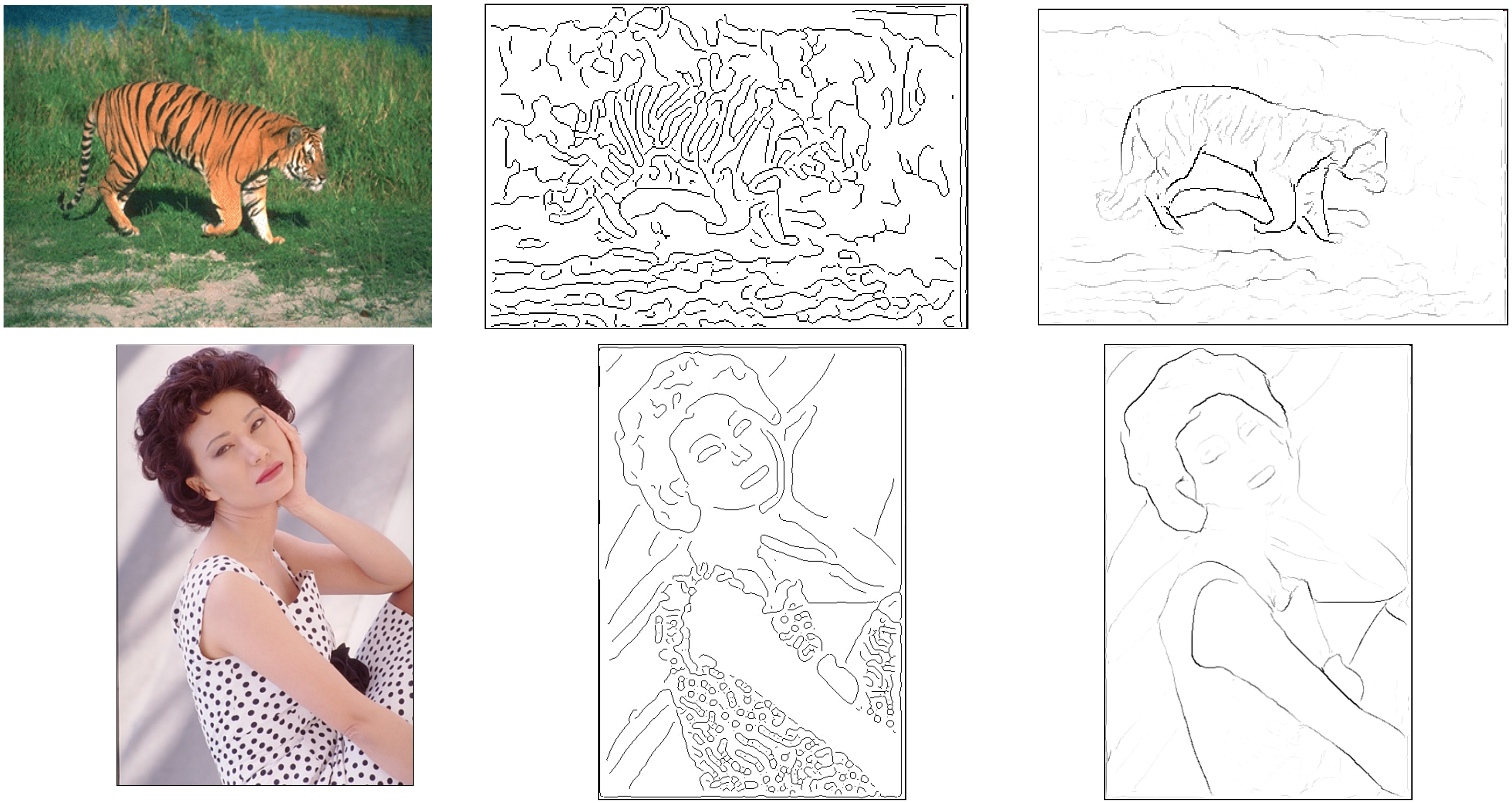
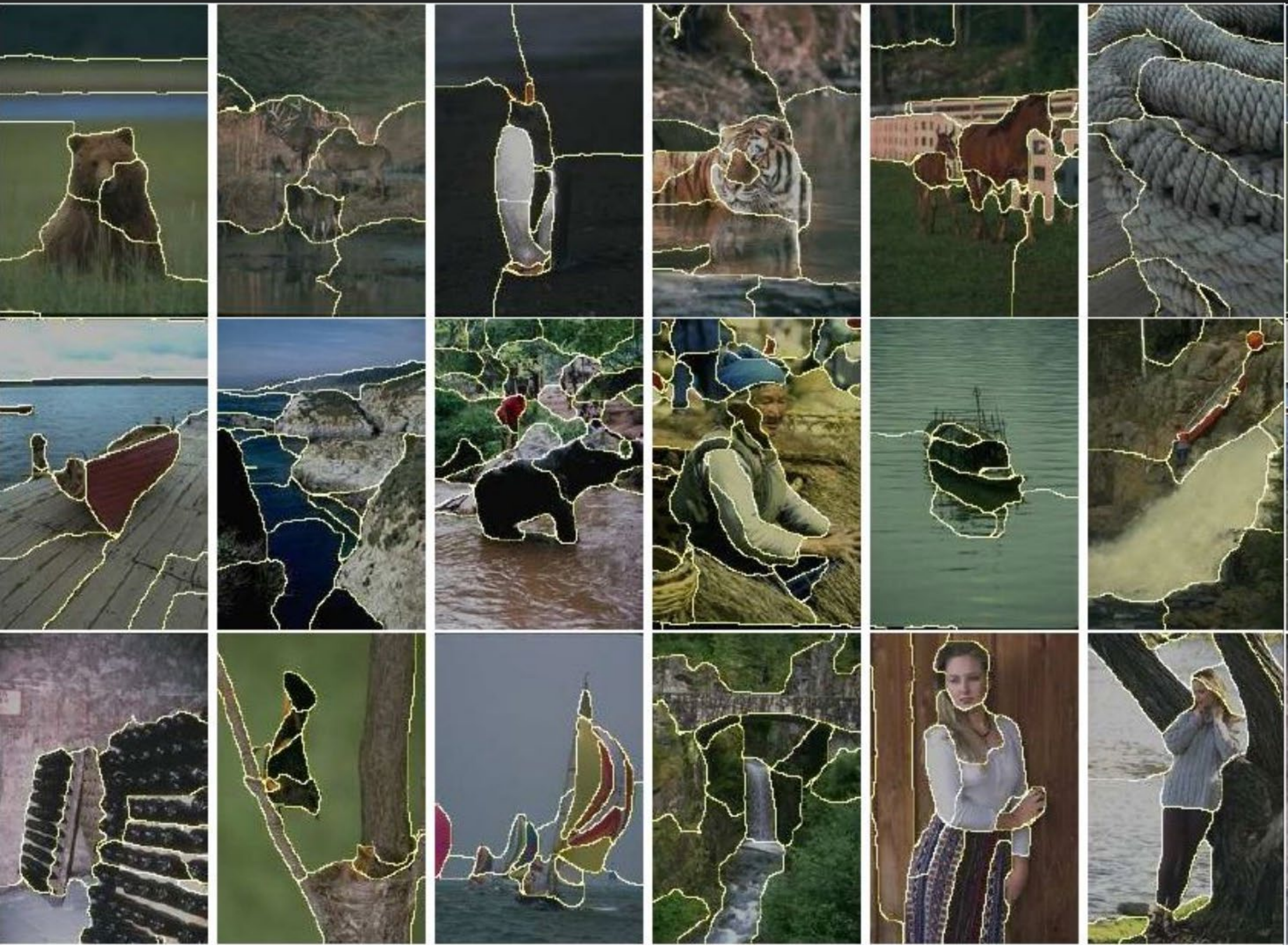


Image Canny Pb

Martin, Fowlkes, Malik, *Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues*, PAMI 2004



Summary

- Segmentation:
  - Partitioning image into coherent regions
- Algorithms:
  - Divisive and hierarchical clustering
  - *k*-means clustering
  - Mean shift clustering
  - Graph cuts
- Applications
  - Image processing, object recognition, interactive image editing, etc.